INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-15 18:59 UTC

node-ipc npm Supply Chain Compromise: Malicious Versions Exfiltrate Cloud, CI/CD, and Container Credentials via DNS Tunneling

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0318
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	node-ipc npm package versions 9.1.6, 9.2.3, 12.0.1; credential targets include AWS, Azure, GCP, OCI, DigitalOcean, Kubernetes, Docker, Helm, Terraform, npm, GitHub, GitLab, Git CLI, macOS Keychain, Firefox, Microsoft Teams
Published	2026-05-15T13:10:42
Discovery Source	Rss

Executive Summary

Three versions of the widely used node-ipc npm package (9.1.6, 9.2.3, 12.0.1) were compromised following an account takeover and are actively stealing credentials from any system that installed them. The malicious code harvests cloud provider keys, CI/CD secrets, container credentials, SSH keys, and browser tokens, then exfiltrates them via DNS tunneling, a method that bypasses most conventional network monitoring. With roughly 690,000 weekly downloads, any organization running Node.js build pipelines, CI/CD systems, or applications with node-ipc as a direct or transitive dependency faces material risk of credential theft and downstream cloud infrastructure compromise.

Technical Analysis

Three malicious versions of the node-ipc npm package were published after an attacker compromised the 'atiertant' npm account (an inactive maintainer). Affected versions: 9.1.6, 9.2.3, and 12.0.1. The embedded infostealer targets an unusually broad credential surface: AWS (~/.aws/credentials, config), Azure (~/.azure/), GCP (~/.config/gcloud/), OCI (~/.oci/), DigitalOcean (~/.config/doctl/), Kubernetes (~/.kube/config), Docker (~/.docker/config.json), Helm (~/.helm/), Terraform (.terraform/ state and variable files), npm (~/.npmrc), GitHub/GitLab tokens, Git CLI credentials (~/.gitconfig, ~/.git-credentials), SSH private keys (~/.ssh/), macOS Keychain entries, Firefox stored credentials, and Microsoft Teams tokens. Exfiltration uses DNS TXT record queries (MITRE T1048.003), a technique that evades security controls focused exclusively on HTTP/HTTPS

egress. Relevant CWEs: CWE-506 (Embedded Malicious Code), CWE-494 (Download of Code Without Integrity Check), CWE-522 (Insufficiently Protected Credentials), CWE-312 (Cleartext Storage of Sensitive Information). MITRE ATT&CK techniques include T1195.002 (Compromise Software Supply Chain), T1083 (File and Directory Discovery), T1555/T1555.003 (Credentials from Password Stores / Browsers), T1552.001 (Credentials in Files), T1552.004 (Private Keys), T1041 (Exfiltration Over C2 Channel), T1048.003 (Exfiltration Over Alternative Protocol, DNS), T1027 (Obfuscated Files), T1078.004 (Valid Cloud Accounts), T1560.001 (Archive Collected Data), T1199 (Trusted Relationship), T1195.001 (Compromise Software Dependencies and Development Tools). No CVE has been assigned as of publication. Safe versions are any release predating 9.1.6 that was not subsequently replaced, or versions published after the malicious versions were removed, verify via npm audit and the package's official advisory.

Action Checklist

1. Containment: immediately run 'npm list node-ipc' across all Node.js projects, CI/CD pipeline dependencies, and container images to identify installations of versions 9.1.6, 9.2.3, or 12.0.1; isolate any build system or runtime that has executed one of these versions pending credential rotation.
2. Detection: query DNS logs for anomalous TXT record lookups originating from build servers, CI/CD runners, or Node.js application hosts; review SIEM for file access events touching ~/.aws/, ~/.kube/config, ~/.docker/config.json, ~/.ssh/, ~/.npmrc, ~/.git-credentials, and equivalent paths on affected hosts; cross-reference with npm install timestamps for the three malicious versions.
3. Eradication: pin node-ipc to a verified clean version in package.json and package-lock.json; run 'npm audit' and 'npm ci' (not 'npm install') to enforce lockfile integrity; remove and rebuild any container images that included the compromised versions; rotate ALL credentials that could have been resident on any affected host (see full credential surface in technical summary).
4. Recovery: after credential rotation, audit cloud provider access logs (AWS CloudTrail, Azure Activity Log, GCP Audit Logs) for unauthorized API calls or resource creation in the window following installation of the malicious version; re-enable CI/CD pipelines only after lockfile verification and image rebuild; monitor DNS logs for continued anomalous TXT queries as a residual indicator.
5. Post-Incident: implement npm package integrity controls: enforce 'npm ci' in all CI/CD pipelines, add a Software Composition Analysis (SCA) tool (e.g., Socket.dev, Snyk, Dependabot) to detect malicious package behavior before installation, restrict npm publish permissions using granular token scoping, and evaluate whether inactive maintainer accounts represent ongoing supply chain risk in your dependency tree.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, legal counsel, and breach notification review immediately if audit logs confirm any exfiltrated credential was used to access systems storing PII, PHI, or PCI data, or if cloud provider logs show unauthorized resource creation, data exfiltration, or IAM privilege escalation using the compromised keys — any of these conditions likely triggers mandatory breach notification under GDPR Article 33, HIPAA §164.400, or applicable state breach laws.

Recovery Notes	Do not re-enable any CI/CD pipeline until: (1) package-lock.json has been regenerated with a verified clean version of node-ipc pinned and the lockfile integrity hash validated against an out-of-band trusted reference, and (2) all credentials confirmed resident on affected hosts during the exposure window have been rotated and the old credentials confirmed revoked in the respective provider console. Monitor AWS CloudTrail, Azure Activity Log, GCP Audit Logs, and DNS TXT query logs continuously for a minimum of 30 days post-rotation, as threat actors who received exfiltrated credentials via DNS tunneling may defer exploitation to avoid detection — delayed use of stolen cloud keys or Kubernetes service account tokens is a documented post-supply-chain-compromise pattern. Retain all forensic artifacts and log exports for a minimum of 12 months to support any regulatory investigation or downstream customer notification obligations.
Forensic Artifacts	DNS resolver logs or pcap filtered for TXT record queries (Wireshark filter: dns.qry.type == 16) from build server and CI/CD runner IPs — the node-ipc malicious payload specifically used DNS TXT records as the exfiltration channel, so base64-encoded subdomains in TXT queries are the primary network-layer indicator of successful credential theft | npm debug and install logs at ~/.npm/_logs/ (Linux/macOS) or %APPDATA%\npm-cache_logs\ (Windows) containing timestamped records of node-ipc 9.1.6, 9.2.3, or 12.0.1 installation, establishing the precise start of the credential exposure window | Linux auditd logs or Windows Security Event Log Event ID 4663 (Attempt to Access an Object) for file read operations targeting ~/.aws/credentials, ~/.kube/config, ~/.docker/config.json, ~/.ssh/id_rsa, ~/.npmrc, ~/.git-credentials, and ~/Library/Keychains/ — these paths are explicitly targeted by the node-ipc malicious payload's credential harvesting routine | Filesystem metadata (mtime, atime, ctime via 'stat') on node_modules/node-ipc/ payload files and the specific JavaScript files containing the credential harvesting and DNS tunneling code — timestamps confirm when the malicious code was written to disk relative to npm install events | Cloud provider API audit trails (AWS CloudTrail event history, GCP Cloud Audit Logs, Azure Activity Log) filtered for the IAM principals associated with credentials resident on affected hosts during the exposure window, specifically for high-value actions: AssumeRole, GetSecretValue, CreateUser, RunInstances, storage.objects.get, Microsoft.KeyVault/vaults/secrets/read — these logs establish whether exfiltrated credentials were subsequently exploited

Per-Action IR Details

Containment — immediately run 'npm list node-ipc' across all Node.js projects, CI/CD pipeline dependencies, and container images to identify installations of versions 9.1.6, 9.2.3, or 12.0.1; isolate any build system or runtime that has executed one of these versions pending credential rotation.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST SI-3 (Malicious Code Protection), NIST CM-2 (Baseline Configuration), CIS 2.3 (Address Unauthorized Software), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Run 'npm list node-ipc 2>/dev/null | grep -E "9\.1\.6|9\.2\.3|12\.0\.1"' recursively across all project directories using: 'find / -name package-lock.json 2>/dev/null | xargs grep -l "node-ipc"'. For container images, run 'docker inspect | grep -i node-ipc' or extract the layer with 'docker save | tar -xO | grep node-ipc'. Network-isolate affected build runners by blocking egress at the host firewall with 'iptables -I OUTPUT -j DROP' or Windows Firewall 'netsh advfirewall set allprofiles firewallpolicy blockinbound,blockoutbound' until credential rotation is complete.

Evidence: Before isolating, capture: (1) full output of 'npm list node-ipc --all --json' to document the exact installed version and dependency chain; (2) process list snapshot ('ps aux' or 'Get-Process') showing any active Node.js processes that loaded node-ipc; (3) network connection state ('ss -tunap' or 'netstat -anob') to capture any live DNS tunneling sessions in progress; (4) filesystem timestamps on node_modules/node-ipc/ directory — specifically mtime on the malicious payload files — using 'stat node_modules/node-ipc/*.js'; (5) container image manifest hashes ('docker

inspect --format="{{.Id}}" ') for all images built after the malicious version publication window.

Detection — query DNS logs for anomalous TXT record lookups originating from build servers, CI/CD runners, or Node.js application hosts; review SIEM for file access events touching ~/.aws/, ~/.kube/config, ~/.docker/config.json, ~/.ssh/, ~/.npmrc, ~/.git-credentials, and equivalent paths on affected hosts; cross-reference with npm install timestamps for the three malicious versions.

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-2 (Event Logging), NIST AU-3 (Content of Audit Records), CIS 8.2 (Collect Audit Logs)

Compensating: Without SIEM: (1) DNS tunneling detection — run 'tcpdump -i any port 53 -w dns_capture.pcap' on build hosts and analyze with Wireshark filter 'dns.qry.type == 16' (TXT record type) to identify base64-encoded credential blobs in query names; (2) File access detection — deploy Sysmon with EventID 11 (FileCreate) and EventID 23 (FileDelete) targeting paths: C:\Users*\.aws\, C:\Users*\.kube\, C:\Users*\.docker\; on Linux use auditd rules: 'auditctl -w /home -p r -k credential_read' for recursive read monitoring; (3) npm install timestamp correlation — parse npm debug log at ~/.npm/_logs/ or %APPDATA%\npm-cache_logs\ filtering for 'node-ipc' entries with timestamps matching versions 9.1.6, 9.2.3, or 12.0.1; (4) use osquery to detect credential file reads: 'SELECT * FROM file_events WHERE path LIKE "/home/%/.aws/%" AND time > '.

Evidence: Capture before analysis: (1) DNS resolver logs or pcap filtered for TXT queries with query names matching base64 patterns (regex: '[A-Za-z0-9+/]{20,}') from the affected host IPs — this is the primary exfiltration channel for this specific campaign; (2) Linux auditd logs (/var/log/audit/audit.log) or Windows Security Event Log Event ID 4663 (Object Access) for read operations on ~/.aws/credentials, ~/.aws/config, ~/.kube/config, ~/.docker/config.json, ~/.ssh/id_rsa, ~/.npmrc, ~/.git-credentials, and macOS Keychain files at ~/Library/Keychains/; (3) npm install logs from ~/.npm/_logs/ documenting the exact install timestamp of node-ipc 9.1.6, 9.2.3, or 12.0.1 to establish the credential exposure window; (4) shell history files (.bash_history, .zsh_history) from CI/CD runner accounts to identify what credentials were in scope during the execution window.

Eradication — pin node-ipc to a verified clean version in package.json and package-lock.json; run 'npm audit' and 'npm ci' (not 'npm install') to enforce lockfile integrity; remove and rebuild any container images that included the compromised versions; rotate ALL credentials that could have been resident on any affected host (see full credential surface in technical summary).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST IA-5 (Authenticator Management), NIST CM-3 (Configuration Change Control), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Credential rotation checklist for teams without secrets management tooling: (1) AWS — 'aws iam delete-access-key --access-key-id && aws iam create-access-key' then update all pipeline environment variables; (2) Kubernetes — revoke and regenerate service account tokens: 'kubectl delete secret -n '; (3) Docker Hub — rotate access tokens at hub.docker.com/settings/security; (4) GitHub/GitLab — revoke all personal access tokens and deploy keys from the accounts whose .git-credentials or token files were resident on affected hosts; (5) npm — revoke publish tokens: 'npm token revoke '; (6) SSH keys — remove compromised public keys from ~/.ssh/authorized_keys on all target systems and generate new keypairs. For container image integrity verification before rebuild, compute SHA256 of the clean base image: 'docker inspect --format="{{index .RepoDigests 0}}" ' and validate against registry digest.

Evidence: Before eradicating, preserve: (1) a read-only snapshot of the malicious node-ipc package files (node_modules/node-ipc/services/ipc.js or equivalent payload file) using 'sha256sum node_modules/node-ipc/**/*.js > malicious_file_hashes.txt' — these hashes constitute forensic evidence of the supply chain compromise; (2) the full package-lock.json before modification, capturing the resolved integrity hash for the malicious version; (3) a memory dump of any Node.js process that executed node-ipc (using 'gcore ' on Linux) if the process is still running — this may contain decrypted credential values in heap; (4) container image layers ('docker save > evidence_image.tar') before rebuilding, as the image history documents exactly which malicious version was embedded and when.

Recovery — after credential rotation, audit cloud provider access logs (AWS CloudTrail, Azure Activity Log, GCP Audit Logs) for unauthorized API calls or resource creation in the window following installation of the malicious version; re-enable CI/CD pipelines only after lockfile verification and image rebuild; monitor DNS logs for continued anomalous TXT queries as a residual indicator.

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), NIST SI-4 (System Monitoring), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without enterprise cloud monitoring: (1) AWS — 'aws cloudtrail lookup-events --lookup-attributes AttributeKey=Username,AttributeValue= --start-time ' filtering for CreateUser, AttachPolicy, RunInstances, CreateBucket, GetSecretValue, and AssumeRole events; (2) GCP — 'gcloud logging read "protoPayload.authenticationInfo.principalEmail= AND timestamp>="' filtering for compute.instances.insert, storage.buckets.create, and secretmanager.versions.access; (3) Azure — query Activity Log via 'az monitor activity-log list --start-time --caller '; (4) For CI/CD pipeline re-enablement gate, use a pre-commit hook or CI step that runs 'npm ci --dry-run' and validates the package-lock.json integrity hash against a known-good reference stored in a separate trusted location; (5) continue DNS TXT query monitoring with tcpdump or network tap for 30 days post-rotation, as DNS tunneling beaconing may continue if any residual credential-bearing process was not identified.

Evidence: Collect before re-enabling pipelines: (1) AWS CloudTrail logs for the period from node-ipc install timestamp to credential rotation completion, specifically filtering event names: GetCallerIdentity, AssumeRole, CreateUser, CreateAccessKey, PutBucketPolicy, GetSecretValue — these represent the highest-value API calls an attacker would make with exfiltrated AWS credentials; (2) Kubernetes API server audit logs (typically at /var/log/kube-apiserver-audit.log) for the same window, filtering for pod creation, secret reads, and cluster-admin binding events; (3) GitHub/GitLab API audit logs for any repository clones, secret reads, or webhook modifications using the compromised tokens; (4) DNS resolver query logs covering the full exposure window — export and retain for minimum 12 months under NIST AU-11 (Audit Record Retention) to support any downstream breach notification requirements.

Post-Incident — implement npm package integrity controls: enforce 'npm ci' in all CI/CD pipelines, add a Software Composition Analysis (SCA) tool (e.g., Socket.dev, Snyk, Dependabot) to detect malicious package behavior before installation, restrict npm publish permissions using granular token scoping, and evaluate whether inactive maintainer accounts represent ongoing supply chain risk in your dependency tree.

NIST Phase: Post Incident

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

Controls: NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST SA-12 (Supply Chain Protection), NIST IR-8 (Incident Response Plan), NIST RA-3 (Risk Assessment), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Free/low-cost supply chain hardening for resource-constrained teams: (1) npm package integrity — add 'npm ci' to every CI job definition and add 'package-lock.json' to branch protection rules requiring PR review before lockfile changes merge; (2) SBOM generation — run 'npm sbom --sbom-format cyclonedx > sbom.json' (available in npm >=v8.8) on each build to document the full dependency tree including transitive dependencies like node-ipc; (3) pre-install behavioral detection — integrate Socket.dev free tier (socket.dev) which specifically detects credential harvesting and DNS tunneling behavior in npm packages before install; (4) maintainer account risk — use 'npm info maintainers' and cross-reference maintainer accounts against HaveIBeenPwned API for known credential breaches, specifically for packages in your dependency tree with >100k weekly downloads; (5) Sigma rule deployment — implement the Sigma rule for DNS TXT record exfiltration (available in SigmaHQ/sigma repository, category: network) tuned to flag queries from build server IP ranges.

Evidence: Preserve for lessons-learned and potential regulatory reporting: (1) complete timeline document mapping node-ipc version publication timestamp → first install on affected host → credential file access events → DNS TXT

exfiltration traffic → credential rotation completion, with all timestamps in UTC; (2) the malicious payload source code from node_modules/node-ipc/ (hash-verified copy) as evidence of the account takeover mechanism and credential targeting logic; (3) full list of all credential types confirmed resident on affected hosts at time of compromise — this list determines breach notification obligations if regulated data (PII, PHI, PCI) was accessible via the exfiltrated cloud or CI/CD credentials; (4) npm registry account access logs for the node-ipc maintainer account showing the account takeover event, requested via npm security team (security@npmjs.com) if not publicly available.

Detection Guidance

Primary detection path: DNS logs. Query for TXT record lookups from build servers, CI/CD runners, or Node.js application hosts to any external domain, especially high-frequency or automated-pattern queries, which are atypical for these systems. Secondary path: file system access monitoring. Look for processes associated with Node.js or npm accessing credential file paths including ~/.aws/credentials, ~/.kube/config, ~/.docker/config.json, ~/.ssh/id_rsa (and variants), ~/.npmrc, ~/.git-credentials, ~/.config/gcloud/application_default_credentials.json, ~/.azure/, and macOS Keychain access events. In CI/CD systems (GitHub Actions, GitLab CI, Jenkins), review job logs for unexpected network activity or file read operations during the build phase. SCA tools with behavioral analysis can provide pre-install detection for future variants. Specific DNS exfiltration domains were not confirmed in public sources at publication. Treat any anomalous TXT-record DNS egress from build/runtime hosts as suspicious pending further analysis.

Indicators of Compromise

Type	Value	Context	Confidence
HASH	`node-ipc@9.1.6`	Malicious npm package version — contains embedded infostealer with DNS exfiltration	HIGH
HASH	`node-ipc@9.2.3`	Malicious npm package version — contains embedded infostealer with DNS exfiltration	HIGH
HASH	`node-ipc@12.0.1`	Malicious npm package version — contains embedded infostealer with DNS exfiltration	HIGH
DOMAIN	`[not confirmed in available sources]`	DNS TXT exfiltration destination domains were not confirmed in T3 sources available at publication — monitor for anomalous TXT record queries from build/runtime hosts as a behavioral IOC	LOW

Framework Mappings

MITRE-ATTACK

- **T1195.002** — Compromise Software Supply Chain
- **T1083** — File and Directory Discovery

- **T1078.004** — Cloud Accounts
- **T1027** — Obfuscated Files or Information
- **T1555** — Credentials from Password Stores
- **T1555.003** — Credentials from Web Browsers
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1552.001** — Credentials In Files
- **T1041** — Exfiltration Over C2 Channel
- **T1199** — Trusted Relationship
- **T1560.001** — Archive via Utility
- **T1552.004** — Private Keys
- **T1048.003** — Exfiltration Over Unencrypted Non-C2 Protocol

NIST-800-53R5

- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **IA-5** — Authenticator Management
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

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

CIS-V8

- **5.2** — Use Unique Passwords
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **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

- **164.312(e)(1)** — Transmission Security

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
T1195.002	Compromise Software Supply Chain	Initial-Access
T1083	File and Directory Discovery	Discovery
T1078.004	Cloud Accounts	Defense-Evasion
T1027	Obfuscated Files or Information	Defense-Evasion
T1555	Credentials from Password Stores	Credential-Access
T1555.003	Credentials from Web Browsers	Credential-Access
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1552.001	Credentials In Files	Credential-Access
T1041	Exfiltration Over C2 Channel	Exfiltration
T1199	Trusted Relationship	Initial-Access
T1560.001	Archive via Utility	Collection
T1552.004	Private Keys	Credential-Access
T1048.003	Exfiltration Over Unencrypted Non-C2 Protocol	Exfiltration

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/popular-node-ipc-npm...	T3
Active Supply Chain Attack: Malicious node-ipc Versions Published ...	https://www.stepsecurity.io/blog/node-ipc-npm-supply-chain-attack	T3
Stealer Backdoor Found in 3 Node-IPC Versions Targeting ...	https://thehackernews.com/2026/05/stealer-backdoor-found-in-3-node-...	T3
Popular node-ipc npm Package Infected with Credential Stealer	https://socket.dev/blog/node-ipc-package-compromised	T3
Compromised node-ipc on npm: Credential Stealer via DNS ...	https://safedep.io/malicious-node-ipc-npm-compromise	T3

Generated 2026-05-15 18:59 UTC by TJS Security Command Center