INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-17 13:57 UTC

Hijacked Contributor Account Used to Poison 144 Mastra npm Packages With Cross-Platform Infostealer

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0499
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	@mastra/core and 143 additional @mastra/* npm packages; easy-day-js npm package; Node.js/npm ecosystem; Windows, macOS, Linux
Published	2026-06-17T03:38:24
Discovery Source	Rss

Executive Summary

On June 17, 2026, attackers hijacked a dormant contributor account for the Mastra AI development framework and published malicious versions of 144 npm packages within 88 minutes, injecting credential and cryptocurrency-stealing malware into a supply chain used by AI and cloud development teams. Any workstation, CI/CD pipeline, or build runner that installed affected @mastra/* package versions during the compromise window must be treated as fully compromised, including all stored credentials, API keys, and cryptocurrency wallet data. With over 918,000 weekly downloads of @mastra/core alone, the blast radius spans development organizations globally, with direct exposure to source code repositories, cloud provider credentials, and internal infrastructure accessible from poisoned build environments.

Technical Analysis

On June 17, 2026, the npm account 'ehindero', a dormant Mastra contributor, was hijacked, likely via credential theft. The attacker (operating as or through 'sergey2016') exploited a provenance policy gap in the Mastra npm organization: personal access token-based publishes were permitted without SLSA attestation or provenance verification. Within an 88-minute window, 144 @mastra/* packages received malicious version publishes injecting 'easy-day-js' as a poisoned dependency. The easy-day-js package delivers a cross-platform infostealer (Windows, macOS, Linux) targeting credentials, API keys, and cryptocurrency wallets commonly present in AI/cloud development environments. Attack chain: T1078/T1078.001 (valid account takeover), T1195.001 (supply chain compromise via dependency injection), T1554 (compromise software supply chain),

T1059/T1059.007 (script execution at install/postinstall), T1105 (payload ingestion), T1140 (deobfuscation), T1555/T1552 (credential and secrets harvesting), T1041/T1071.001 (exfiltration over C2 channel), T1547 (persistence establishment), T1036.005 (masquerading via legitimate package name). CWEs: CWE-506 (embedded malicious code), CWE-269 (privilege abuse via hijacked contributor role), CWE-494 (missing integrity verification on published packages), CWE-295 (absent attestation/certificate validation). No CVE has been assigned. Affected packages: @mastra/core and 143 additional @mastra/* packages; the malicious 'easy-day-js' package is the delivery vehicle. Safe versions are those published before June 17, 2026 or after Mastra confirmed remediation, verify specific version ranges against the Endor Labs advisory. Confirm affected version ranges and remediation status directly with Mastra's official npm organization and the Endor Labs report before acting.

Action Checklist

1. Step 1: Containment, Immediately audit all package-lock.json, yarn.lock, and pnpm-lock.yaml files across development workstations, CI runners, and build pipelines for any @mastra/* package versions published on June 17, 2026 (within or near the 88-minute compromise window). Block installation of 'easy-day-js' at your package registry proxy or internal npm mirror. Isolate any host that installed affected versions during the window, treat them as fully compromised and do not allow continued access to production systems or secrets stores.
2. Step 2: Detection, Search CI/CD logs and npm audit logs for installs of @mastra/* packages on June 17, 2026. Query your SIEM for outbound connections from build agents and developer workstations to unknown or anomalous external hosts (T1041/T1071.001 exfiltration indicators). Review endpoint logs for postinstall script execution events (T1059/T1059.007) tied to node or npm processes. Check for new persistence mechanisms on affected hosts: scheduled tasks, startup entries, or modified shell profiles (T1547). Indicators of compromise (see iocs field) should be used to hunt across DNS, proxy, and EDR telemetry. Reference NIST AU-6 (audit record review) and CIS 8.2 (audit log collection) to confirm log coverage exists before declaring clean.
3. Step 3: Eradication, Pin all @mastra/* dependencies to versions confirmed safe by the Mastra maintainers or the Endor Labs advisory. Verify version ranges against official sources rather than relying on npm audit alone, as malicious versions may have been unpublished from npm's public registry by the time you audit. Remove 'easy-day-js' from all dependency trees. Rotate all secrets, API keys, cloud provider credentials, and cryptocurrency wallet keys accessible from any compromised host or CI environment. Revoke and reissue any tokens stored in environment variables, .env files, or secrets managers reachable from affected build environments. Apply CIS 5.3 (disable dormant accounts) controls to your own npm organization publish permissions immediately.
4. Step 4: Recovery, Before returning any isolated host to production use, reimage or perform a clean OS reinstall, do not attempt to clean in place given the cross-platform stealer's persistence capabilities (T1547). Rebuild CI runners from verified base images. Re-audit your internal npm mirror or registry proxy to confirm malicious package versions have been purged. Restore secrets from backup only after confirmed rotation. Monitor for anomalous authentication events (NIST AC-7, AC-2) using rotated credentials as canaries for the next 30 days. Validate SLSA provenance attestation is enforced before re-enabling @mastra/* package consumption.
5. Step 5: Post-Incident, This attack exploited a provenance policy gap (CWE-494, CWE-295): the Mastra npm organization permitted personal token publishes without SLSA attestation. Implement mandatory provenance attestation (SLSA Level 2 or higher) for all packages in your own npm organizations. Enforce

npm package signing verification in your CI pipelines. Audit dormant contributor accounts across all package registries your organization maintains or consumes (CIS 5.3, NIST AC-2). Implement a private registry proxy with allowlist controls to intercept unexpected new dependencies before they execute in builds (D3-PBWSAM as an analog for registry mediation). Review secrets management to ensure CI environment variables are scoped narrowly and rotated on a schedule (NIST AC-6, D3-CRO). Conduct a full software supply chain risk review against NIST SP 800-161r1 supply chain risk management practices.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, legal counsel, and breach notification workflows immediately if forensic analysis of network egress logs or cloud provider audit trails (AWS CloudTrail, GCP Audit Logs, Azure Activity Log) confirms that credentials exfiltrated from CI environments were used to access systems containing PII, PHI, PCI-scoped data, or if the compromised build pipeline had production deployment privileges — any confirmed attacker use of stolen credentials triggers breach notification assessment under GDPR Article 33, CCPA, or applicable state breach notification law.
Recovery Notes	No affected host should be returned to production without a verified clean OS reinstall from a pre-incident golden image — the cross-platform persistence mechanisms documented for this stealer class (shell profile modification, scheduled tasks, launchd plists, Windows Run keys) make in-place cleanup unreliable across Windows, macOS, and Linux. All rotated credentials should be treated as active canaries for a minimum of 30 days: any authentication event using a pre-rotation credential after confirmed rotation is a high-confidence indicator of ongoing attacker access or credential reuse and must trigger immediate incident re-escalation. Before re-enabling @mastra/* package consumption, validate that the package versions in use carry SLSA provenance attestation verifiable via `npm install --audit-signatures` and that your internal registry proxy has been updated to enforce attestation checks at install time.

Forensic Artifacts	npm install debug logs at `~/.npm/_logs/*.log` (Linux/macOS) or `%AppData%\npm-cache\_logs\*.log` (Windows) containing the resolved @mastra/* package version, registry URL, and postinstall script execution timestamp for June 17, 2026 — these are the primary evidence of which specific malicious package version was pulled and when the postinstall stealer payload executed | Process creation records (Windows Security Event ID 4688 with command-line auditing enabled, or Sysmon Event ID 1) for `node.exe` / `node` processes spawned by `npm` during the install window, including the full command line of any child process that indicates encoded payload execution (base64 arguments, PowerShell -EncodedCommand, curl/wget to non-npm registry hosts) | Network proxy and firewall egress logs for CI runner and developer workstation network segments covering June 17, 2026, filtered for outbound HTTPS or DNS from `node` process — the infostealer postinstall payload would have established a C2 or exfil connection immediately upon package execution, making this the highest-value artifact for determining whether data left the environment | Filesystem timeline artifacts showing files created or modified by the Node.js process after the malicious @mastra/* install: specifically check `$TEMP` / `/tmp` for dropped binaries or scripts, `~/.ssh/` for added authorized keys or modified known_hosts, and `~/.aws/credentials`, `~/.config/gcloud/`, and `~/.azure/` for credential file access timestamps consistent with stealer exfiltration activity | Cloud provider audit logs (AWS CloudTrail `LookupEvents` filtered on `sourceIPAddress` of affected CI hosts, GCP `gcloud logging read` for the CI service account, Azure Monitor activity log) for the period June 17 through current date — these are the definitive record of whether any credentials harvested by the postinstall stealer were subsequently used by the attacker to access cloud infrastructure, and are essential for scope determination and breach notification assessment

Per-Action IR Details

Step 1: Containment — Immediately audit all package-lock.json, yarn.lock, and pnpm-lock.yaml files across development workstations, CI runners, and build pipelines for any @mastra/* package versions published on June 17, 2026 (within or near the 88-minute compromise window). Block installation of 'easy-day-js' at your package registry proxy or internal npm mirror. Isolate any host that installed affected versions during the window — treat them as fully compromised and do not allow continued access to production systems or secrets stores.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: Choose a containment strategy based on the type of incident; criteria include potential damage, need for evidence preservation, and service availability. For supply chain compromise with active credential theft, immediate network isolation of affected hosts is required before any eradication action.

Controls: NIST AC-4 (Information Flow Enforcement), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 2.1 (Establish and Maintain a Software Inventory)

Compensating: For teams without a centralized SIEM or artifact scanning platform: run `find . -name 'package-lock.json' -o -name 'yarn.lock' -o -name 'pnpm-lock.yaml' | xargs grep -l '@mastra/'` across all developer workstations and CI runner home directories. For easy-day-js, add a `.npmrc` deny rule (`@easy-day-js:registry=http://localhost:0`) on internal Verdaccio or Nexus mirrors. Use `iptables -I OUTPUT -d 0.0.0.0/0 -m owner --uid-owner $(id -u ci-runner) -j DROP` on Linux CI hosts to hard-isolate runners without a firewall appliance. On Windows, use `netsh advfirewall firewall add rule name='CI-ISOLATE' dir=out action=block` scoped to the build agent service account.

Evidence: Before isolating any host, capture: (1) full RAM dump using WinPmem (Windows) or `avml` (Linux) to preserve any decrypted credentials or C2 beacons resident in the Node.js process heap from the postinstall stealer payload; (2) `netstat -ano` / `ss -tunap` output to record active outbound connections established by the malicious postinstall script at time of install; (3) complete npm install log at `~/.npm/_logs/` (Linux/macOS) or `%AppData%\npm-cache\_logs\` (Windows) showing the exact @mastra/* package versions and timestamps resolved

during the June 17 window; (4) process tree snapshot (`Get-WmiObject Win32_Process` or `ps auxf`) to record any child processes spawned by `node` during postinstall execution; (5) copy of all lock files before any `npm install` or dependency update is run, as re-running npm may overwrite resolved version history.

Step 2: Detection — Search CI/CD logs and npm audit logs for installs of @mastra/* packages on June 17, 2026. Query your SIEM for outbound connections from build agents and developer workstations to unknown or anomalous external hosts (T1041/T1071.001 exfiltration indicators). Review endpoint logs for postinstall script execution events (T1059/T1059.007) tied to node or npm processes. Check for new persistence mechanisms on affected hosts: scheduled tasks, startup entries, or modified shell profiles (T1547). Indicators of compromise (see iocs field) should be used to hunt across DNS, proxy, and EDR telemetry. Reference NIST AU-6 (audit record review) and CIS 8.2 (audit log collection) to confirm log coverage exists before declaring clean.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: Analyze all available precursors and indicators, including those from logs and monitoring systems. Correlate indicators across sources to determine scope; for this campaign, scope determination requires correlating npm install timestamps against network egress and postinstall script execution events across all affected host types.

Controls: 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 a SIEM: query CI/CD pipeline logs manually using `grep -r 'mastra' ~/.npm/_logs/ /var/log/jenkins/ /home/runner/` for install timestamps on June 17, 2026. Use Sysmon Event ID 1 (Process Creation) filtered on `ParentImage` containing `npm` or `node` and `CommandLine` containing `postinstall` to detect script execution — deploy a Sysmon config with the SwiftOnSecurity baseline if not already present. For network hunting, run `Wireshark` or `tcpdump -w capture.pcap` on the CI segment and filter for outbound connections from `node.exe` or `node` process to non-RFC1918 addresses post-install. Use the Sigma rule `proc_creation_win_node_postinstall_execution` community rule as a starting point for log-based detection. On Linux, check `~/.bash_profile`, `~/.zshrc`, and `/etc/cron.d/` for additions timestamped June 17, 2026 using `find / -newer /tmp/ref_june17 -type f 2>/dev/null` where ref_june17 is a file `touch`-ed to that date.

Evidence: Before taking any remediation action on a potentially live host: (1) collect all npm debug logs from `~/.npm/_logs/*.log` and CI system logs (GitHub Actions: `.github/workflows/` run logs; Jenkins: `$JENKINS_HOME/jobs/*/builds/*/log`) timestamped June 17, 2026 showing resolved @mastra/* package versions; (2) capture DNS query logs from the build network for the compromise window — the stealer postinstall script would have resolved C2 or exfil domains immediately upon package execution; (3) collect Windows Security Event Log Event ID 4688 (Process Creation) or Linux `auditd` execve records for `node` and `npm` processes spawned during the install window, specifically looking for child processes with encoded PowerShell, `curl`, `wget`, or base64 command arguments indicative of the credential-harvesting payload; (4) snapshot all environment variables accessible to the build agent at time of install (`printenv` output or `Get-ChildItem Env:`) since the stealer targets CI environment variables containing API keys and cloud credentials; (5) collect proxy/firewall egress logs for the CI network segment for June 17, 2026 to identify exfiltration destinations before those logs are rotated.

Step 3: Eradication — Pin all @mastra/* dependencies to versions confirmed safe by the Mastra maintainers or the Endor Labs advisory — do not rely solely on 'npm audit' as the malicious versions may have been unpublished by the time you run it. Remove 'easy-day-js' from all dependency trees. Rotate all secrets, API keys, cloud provider credentials, and cryptocurrency wallet keys accessible from any compromised host or CI environment. Revoke and reissue any tokens stored in environment variables, .env files, or secrets managers reachable from affected build environments. Apply CIS 5.3 (disable dormant accounts) controls to your own npm organization publish permissions immediately.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication: After containing the incident, eradicate the cause; this includes deleting malware, disabling breached accounts, and identifying and mitigating all vulnerabilities that were exploited. For

this supply chain campaign, eradication spans both the malicious package artifacts and the credential material exfiltrated by the postinstall stealer.

Controls: NIST AC-2 (Account Management), NIST AC-6 (Least Privilege), CIS 5.3 (Disable Dormant Accounts), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Without enterprise secrets management tooling: enumerate all `.env` files, `~/.aws/credentials`, `~/.config/gcloud/`, `~/.azure/`, and CI environment variable exports using `find / -name '.env' -o -name 'credentials' 2>/dev/null` on each affected host before rotation. Use the AWS CLI `aws iam list-access-keys --user-name ` and `aws iam delete-access-key` to revoke AWS keys directly. For GitHub tokens, use `gh auth status` to enumerate active tokens and revoke via the GitHub UI under Settings > Developer Settings > Personal Access Tokens. For npm organization publish permissions, run `npm access ls-collaborators @mastra` to enumerate and `npm access revoke` to remove dormant contributor tokens — do not wait for the npm registry to act. Document each credential revoked with timestamp for the post-incident record.

Evidence: Before rotating any credential or revoking any token, capture: (1) a complete list of all environment variables available to each CI runner at the time of the June 17 build — these represent the exact secrets the postinstall stealer would have targeted for exfiltration; (2) AWS CloudTrail / GCP Audit Logs / Azure Activity Logs for API calls made using the potentially-compromised CI credentials in the 24–72 hours following June 17, to establish whether exfiltrated keys were actually used by the attacker before rotation; (3) contents of any `.env` files, secrets manager secret metadata (not values), and CI/CD pipeline variable listings (do not print plaintext secrets into logs — record only secret names and last-rotated dates); (4) npm publish access token audit from `npm token list` showing all active tokens for the organization before revocation, to confirm whether the hijacked contributor pattern used in the Mastra attack also exists in your own npm org.

Step 4: Recovery — Before returning any isolated host to production use, reimage or perform a clean OS reinstall — do not attempt to clean in place given the cross-platform stealer's persistence capabilities (T1547). Rebuild CI runners from verified base images. Re-audit your internal npm mirror or registry proxy to confirm malicious package versions have been purged. Restore secrets from backup only after confirmed rotation. Monitor for anomalous authentication events (NIST AC-7, AC-2) using rotated credentials as canaries for the next 30 days. Validate SLSA provenance attestation is enforced before re-enabling @mastra/* package consumption.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery: Restore systems to normal operation, confirm that the systems are functioning normally, and implement additional monitoring to watch for recurrence. For hosts compromised by the Mastra cross-platform stealer, in-place remediation cannot be trusted given the stealer's documented cross-platform persistence — clean reinstall is the minimum acceptable recovery bar.

Controls: NIST AC-2 (Account Management), NIST AC-7 (Unsuccessful Logon Attempts), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 7.3 (Perform Automated Operating System Patch Management)

Compensating: Without commercial image management: use a documented and version-controlled Dockerfile or Packer template stored in a separate, unaffected repository as the canonical CI runner base image — rebuild from that template rather than restoring from a snapshot taken after June 17. Verify the base image hash against the registry digest before use. For developer workstation reimaging on a budget, use a network boot (PXE/WDS) golden image with a known-good hash recorded before the incident date. For the 30-day canary monitoring without a SIEM, configure AWS CloudWatch Alarms or free-tier Datadog agent alerts on any API call from the rotated CI credentials, and enable GitHub audit log streaming to an S3 bucket for review. Use `last` (Linux) and Windows Security Event ID 4624/4625 log exports to track authentication events against the new credentials.

Evidence: Before reimaging any host, ensure forensic evidence capture (RAM, disk image, volatile network state) is complete and preserved to an isolated evidence store — reimaging destroys all persistence artifacts the cross-platform stealer installed (modified shell profiles, scheduled tasks, launchd plists on macOS, systemd units on Linux, Windows registry Run keys). Specifically document: (1) disk image or at minimum file system timeline (`fls` via Sleuth Kit or `MFTECmd` on Windows) showing files created or modified on or after June 17, 2026 by the Node.js process; (2) macOS LaunchAgent/LaunchDaemon plist files in `~/Library/LaunchAgents/` and `/Library/LaunchDaemons/` if macOS

hosts are in scope; (3) Windows registry export of `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` and `HKLM\Software\Microsoft\Windows\CurrentVersion\Run` showing any persistence entries added by the stealer postinstall payload.

Step 5: Post-Incident — This attack exploited a provenance policy gap (CWE-494, CWE-295): the Mastra npm organization permitted personal token publishes without SLSA attestation. Implement mandatory provenance attestation (SLSA Level 2 or higher) for all packages in your own npm organizations. Enforce npm package signing verification in your CI pipelines. Audit dormant contributor accounts across all package registries your organization maintains or consumes (CIS 5.3, NIST AC-2). Implement a private registry proxy with allowlist controls to intercept unexpected new dependencies before they execute in builds (D3-PBWSAM as an analog for registry mediation). Review secrets management to ensure CI environment variables are scoped narrowly and rotated on a schedule (NIST AC-6, D3-CRO). Conduct a full software supply chain risk review against NIST SP 800-161r1 supply chain risk management practices.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: Review the incident to identify lessons learned, improve policies and procedures, and share threat intelligence. The Mastra campaign demonstrates a class of supply chain risk — dormant contributor account hijack enabling package poisoning — that requires structural policy changes, not just tactical remediation, to prevent recurrence.

Controls: NIST AC-2 (Account Management), NIST AC-6 (Least Privilege), NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 5.3 (Disable Dormant Accounts), CIS 6.2 (Establish an Access Revoking Process), CIS 6.5 (Require MFA for Administrative Access), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without enterprise tooling for supply chain governance: (1) enumerate dormant npm publish tokens using `npm token list` for every package your organization owns and revoke any token unused for 90+ days; (2) enforce npm org-level 2FA requirement via `npm org set-2fa-required ` — this is free and directly addresses the hijacked contributor account vector used in this attack; (3) deploy Verdaccio (free, open source) as a private npm proxy with an explicit package allowlist — any package not on the allowlist is blocked before `postinstall` scripts can execute; (4) add a `package.json` pre-install hook using `npm-audit-resolver` or a custom `preinstall` script that validates `npm pack --dry-run` integrity hashes against a pinned manifest; (5) configure Dependabot or Renovate (both free tiers available) with lock-file-only mode to prevent unexpected transitive dependency version changes from pulling in new publisher versions.

Evidence: For the lessons-learned review, preserve and analyze: (1) the complete timeline of @mastra/* package version publishes on June 17, 2026 reconstructed from npm registry metadata (`npm view @mastra/core time --json`) — this establishes the 88-minute window precisely for policy gap analysis; (2) a full inventory of which internal projects, CI pipelines, and developer workstations had @mastra/* in their dependency trees before the incident, derived from the lock files collected during containment; (3) documentation of all secrets and credentials in scope for the CI environments affected, with evidence of rotation completion dates, for regulatory and audit purposes; (4) network egress records showing whether any confirmed exfiltration occurred and to which destinations, to determine breach notification obligations if PII or regulated data was accessible from the compromised CI environments.

Detection Guidance

Primary hunt: search package manager logs and CI/CD build logs for any install of @mastra/* packages on June 17, 2026, and for any version of 'easy-day-js' in dependency trees at any time. Secondary hunt: query proxy/DNS logs for outbound connections from build agents or developer workstations to external hosts not in your approved egress list, specifically during and after June 17, 2026 build jobs. Behavioral indicators: unexpected postinstall script execution by node or npm processes (T1059.007); new scheduled tasks, cron entries, or shell profile modifications created by node processes (T1547); outbound HTTP/HTTPS traffic from build environments to unrecognized hosts (T1071.001); access to credential stores, .env files, or cloud metadata

endpoints from npm child processes (T1552, T1555). EDR query focus: parent process = node.exe or npm spawning network connections or writing to persistence locations. SIEM correlation: join npm install events with subsequent outbound network flows from the same host within a 10-minute window. Reference NIST SI-4 (system monitoring) and CIS 8.2 (audit log collection) to confirm telemetry coverage. NIST AU-6 (audit record review) procedures should be invoked for affected build environments. Specific C2 infrastructure IOCs are not confirmed in available sources; treat any anomalous outbound connection from a build host on June 17, 2026 as suspicious pending further analysis.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`easy-day-js (npm package name)`	Malicious npm package injected as poisoned dependency into all 144 compromised @mastra/* packages; presence in any dependency tree indicates exposure	HIGH
HASH	`not confirmed in available sources`	Specific file hashes for the easy-day-js payload were not confirmed in T3-tier sources available at time of writing; check Endor Labs advisory for updated hash IOCs	LOW
URL	`not confirmed in available sources`	C2 exfiltration endpoint URLs were not confirmed in available sources; hunt for anomalous outbound HTTP/HTTPS from build environments on June 17, 2026 as a proxy indicator	LOW
DOMAIN	`not confirmed in available sources`	C2 domain infrastructure was not confirmed in T3-tier sources; Endor Labs and StepSecurity advisories should be checked for updated network IOCs	LOW

Framework Mappings

MITRE-ATTACK

- **T1078** — Valid Accounts
- **T1059.007** — JavaScript
- **T1554** — Compromise Host Software Binary
- **T1555** — Credentials from Password Stores
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1041** — Exfiltration Over C2 Channel
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1552** — Unsecured Credentials
- **T1059** — Command and Scripting Interpreter

- **T1078.001** — Default Accounts
- **T1140** — Deobfuscate/Decode Files or Information
- **T1105** — Ingress Tool Transfer
- **T1547** — Boot or Logon Autostart Execution
- **T1071.001** — Web Protocols

NIST-800-53R5

- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-4** — System Monitoring
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **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
- **SC-13** — Cryptographic Protection

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control
- **A08:2021** — Software and Data Integrity Failures
- **A02:2021** — Cryptographic Failures
- **A07:2021** — Identification and Authentication Failures

CIS-V8

- **5.4** — Restrict Administrator Privileges to Dedicated Administrator Accounts
- **6.8** — Define and Maintain Role-Based Access Control
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **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

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

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
T1078	Valid Accounts	Defense-Evasion
T1059.007	JavaScript	Execution
T1554	Compromise Host Software Binary	Persistence
T1555	Credentials from Password Stores	Credential-Access
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1041	Exfiltration Over C2 Channel	Exfiltration
T1036.005	Match Legitimate Resource Name or Location	Defense-Evasion
T1552	Unsecured Credentials	Credential-Access
T1059	Command and Scripting Interpreter	Execution
T1078.001	Default Accounts	Defense-Evasion
T1140	Deobfuscate/Decode Files or Information	Defense-Evasion
T1105	Ingress Tool Transfer	Command-And-Control
T1547	Boot or Logon Autostart Execution	Persistence
T1071.001	Web Protocols	Command-And-Control

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/06/144-mastra-npm-packages-compromis ...	T3

Source	URL	Tier
Mastra npm Org Compromised: Multiple Packages Trojanized to ...	https://www.endorlabs.com/learn/mastra-npm-org-compromised-multiple...	T3
Mass Supply Chain Attack Hits TanStack, Mistral AI npm and PyPI ...	https://safedep.io/mass-npm-supply-chain-attack-tanstack-mistral	T3
How can I know that the a library or package in npm is not malicious?	https://www.reddit.com/r/reactjs/comments/15t99e0/how_can_i_know_th...	T3
StepSecurity Blog | GitHub Actions Security Insights	https://www.stepsecurity.io/blog	T3

Generated 2026-06-17 13:57 UTC by TJS Security Command Center