INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-18 07:12 UTC

Mastra npm Supply Chain Compromise: Account Takeover Enables 140+ Package Poisoning via easy-day-js Postinstall Dropper

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0504
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	npm registry (@mastra scope, 140+ packages), Node.js developer environments, Windows/macOS/Linux developer workstations, CI/CD pipelines
Published	2026-06-18T03:43:04+00:00
Discovery Source	Rss:T1 Threatintel

Executive Summary

An unknown attacker hijacked the npm maintainer account for the Mastra AI framework and published poisoned versions of 140+ packages that silently install a remote access trojan and credential-stealing malware during routine dependency installation. Any developer workstation or CI/CD pipeline that ran npm install or npm update against @mastra packages after June 17, 2026 at 01:01 UTC must be treated as fully compromised. The business risk spans source code theft, credential exfiltration, cryptocurrency wallet drain, and lateral movement from developer environments into production systems.

Technical Analysis

Attacker compromised the 'ehindero' npm maintainer account (associated with ehindero2016@tutamail[.]com and sergey2016@tutamail[.]com) and injected a dependency on the typosquatted package 'easy-day-js' across 140+ packages in the @mastra scope. The postinstall hook executes a multi-stage payload at install time: (1) TLS certificate verification is disabled (CWE-295), bypassing HTTPS inspection; (2) a cross-platform remote access trojan with cryptocurrency-stealing capability is fetched over C2 (T1071.001, T1041); (3) platform-specific persistence is established via Windows registry run keys (T1547.001), macOS launch agents/daemons (T1543.001/T1543.004), and Linux equivalents (T1546); (4) reflective .NET assembly injection is performed on Windows hosts (T1055.001, CWE-506). The attack exploits a compromised valid account (T1078, T1078.004) rather than a registry vulnerability, so no CVE is assigned to the compromise vector. Applicable CWEs include CWE-506 (embedded malicious code), CWE-284 (improper access control),

CWE-494 (download of code without integrity check), CWE-693 (protection mechanism failure), and CWE-295 (improper certificate validation). The payload executes before any application code imports, meaning sandboxing at the application layer provides no protection. Microsoft Defender Antivirus, Defender for Endpoint, and Defender XDR have reported detection coverage. No CVE is assigned; CVSS base is assessed at 9.5 (Critical). Attack is classified as supply chain compromise (T1195.001) with obfuscated files (T1027) and credential access (T1552, T1552.001). Affected: npm registry @mastra scope, Node.js developer environments, Windows/macOS/Linux workstations, and CI/CD pipelines.

Action Checklist

1. Step 1: Containment. Immediately identify all developer workstations, CI/CD runners, and build servers that executed 'npm install' or 'npm update' involving any @mastra scoped package after June 17, 2026 at 01:01 UTC. Isolate those systems from the network pending forensic review. Block outbound connections to Tutamail infrastructure (ehindero2016@tutamail[.]com, sergey2016@tutamail[.]com domains) and any unrecognized external IPs spawned by Node.js or dotnet processes. Remove the 'easy-day-js' package from any npm lockfiles, package.json files, and local caches (npm cache clean --force).
2. Step 2: Detection. Query endpoint logs for postinstall script execution originating from npm processes; look for child processes spawned by node.exe or npm.cmd making outbound HTTP/HTTPS connections (especially with TLS verification disabled). On Windows, inspect Defender for Endpoint alerts for reflective .NET injection (T1055.001) and new run key entries under HKCU\Software\Microsoft\Windows\CurrentVersion\Run. On macOS, check ~/Library/LaunchAgents and /Library/LaunchDaemons for unfamiliar plist entries created after June 17, 2026. On Linux, audit cron, systemd unit files, and ~/.bashrc or ~/.profile for injected persistence. Search npm audit logs and CI/CD pipeline logs for any reference to 'easy-day-js' as a resolved dependency. Enable or review Microsoft Defender XDR alerts for the reported detection signatures.
3. Step 3: Eradication. Remove all @mastra packages from affected environments and purge node_modules directories entirely; do not simply reinstall without version pinning to a verified clean release. Remove 'easy-day-js' and any transitive dependencies it introduced. Delete all identified persistence mechanisms: registry run keys (Windows), launch agent/daemon plists (macOS), and systemd/cron entries (Linux). Rotate all credentials, API keys, tokens, and SSH keys present on or accessible from compromised systems, including secrets stored in environment variables, .env files, and CI/CD secret stores (T1552.001 exposure). Revoke and reissue code-signing certificates if present on affected build systems.
4. Step 4: Recovery. Rebuild compromised developer workstations and CI/CD runners from trusted golden images rather than attempting in-place remediation, given the RAT's persistence capability. Re-establish npm dependencies from a clean lockfile with all packages pinned to verified versions and validated against known-good checksums. Re-run builds in an isolated environment and validate build artifact integrity before promoting to production. Monitor rebuilt systems for 30 days using EDR telemetry for signs of reinfection or residual C2 beaconing. Confirm Microsoft Defender detections are resolving clean on rebuilt systems.
5. Step 5: Post-Incident. Conduct a control gap review against NIST SI-4 (system monitoring), NIST CM-7 (least functionality for build systems), and CIS 2.1/2.3 (software inventory and unauthorized software removal). Implement npm package integrity verification using lockfile pinning and Subresource Integrity or equivalent checksum validation (addresses CWE-494, no mapped control for npm-specific supply chain integrity in the knowledge base beyond CIS 7.1/7.2 vulnerability management process). Evaluate adoption

of a private npm registry proxy with allowlisting to prevent direct public registry pulls in CI/CD. Require MFA on all npm maintainer accounts your organization controls (CIS 6.5). Establish a process for monitoring npm account takeover advisories through the npm security advisories feed.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, legal counsel, and breach notification counsel immediately if forensic review confirms that CI/CD runners with access to production secrets, customer data repositories, or code-signing infrastructure executed npm install against any @mastra package after June 17, 2026 01:01 UTC, as credential exfiltration scope may trigger regulatory breach notification obligations under applicable data protection law.
Recovery Notes	Rebuild all confirmed-affected systems from pre-June-17-2026 golden images — do not attempt in-place remediation of hosts where the RAT's .NET reflective injection ran, as persistence mechanisms may survive partial cleanup. Validate all npm package checksums using 'npm audit signatures' and 'npm ci' lockfile enforcement before allowing any rebuilt CI/CD runner to produce artifacts destined for production. Monitor rebuilt systems for a minimum of 30 days with Sysmon Event ID 3 (Network Connection) alerts on outbound connections from node.exe and dotnet.exe, specifically watching for any reconnection attempts to Tutamail infrastructure or previously unresolved IP endpoints, which would indicate a persistence mechanism survived the reimage.
Forensic Artifacts	npm debug logs at ~/.npm/_logs/ (Linux/macOS) and %APPDATA%\npm-cache\ (Windows) — contain timestamped postinstall script execution records that will show easy-day-js dropper invocation during the @mastra package install event, with UTC timestamps correlatable to the June 17, 2026 01:01 UTC compromise window | Windows Sysmon Event ID 1 (Process Create) logs showing node.exe or npm.cmd parent processes spawning cmd.exe, powershell.exe, or dotnet.exe child processes — the reflective .NET injection dropper would produce a distinctive process ancestry chain not present in legitimate npm postinstall hooks | HKCU\Software\Microsoft\Windows\CurrentVersion\Run registry hive — the RAT's Windows persistence mechanism; export and examine for entries created after June 17, 2026 01:01 UTC with paths pointing into node_modules or temp directories | macOS ~/Library/LaunchAgents/ and Linux ~/.bashrc / ~/.profile / systemd unit files — cross-platform persistence artifacts written by the easy-day-js dropper's postinstall hook; file creation timestamps in the June 17, 2026 UTC window are definitive indicators of compromise | CI/CD pipeline execution logs and artifact hashes for all builds run on compromised runners — these logs establish which build outputs may carry injected code from the RAT, enabling a poisoned artifact recall scope assessment; preserve before pipeline log retention policies overwrite them

Per-Action IR Details

Step 1: Containment — Immediately identify all developer workstations, CI/CD runners, and build servers that executed 'npm install' or 'npm update' involving any @mastra scoped package after June 17, 2026 at 01:01 UTC. Isolate those systems from the network pending forensic review. Block outbound connections to Tutamail infrastructure (ehindero2016@tutamail[.]com, sergey2016@tutamail[.]com domains) and any unrecognized external IPs spawned by Node.js or dotnet processes. Remove the 'easy-day-js' package from any npm lockfiles, package.json files, and local caches (npm cache clean --force).

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: Query npm debug logs at ~/.npm/_logs/ and CI/CD pipeline stdout archives for 'easy-day-js' install references. Use PowerShell: Get-Content $env:APPDATA\npm\npm-debug.log* | Select-String 'easy-day-js' (Windows) or grep -r 'easy-day-js' ~/.npm/_logs/ (macOS/Linux). For network blocking without enterprise firewall management, add host-based rules via Windows Defender Firewall netsh advfirewall or iptables/ufw rules blocking outbound to tutamail.com on ports 443 and 80. Disable the affected workstation's NIC via device manager or 'ip link set eth0 down' as emergency isolation before formal firewall changes.

Evidence: BEFORE isolating any host or purging npm caches, capture: (1) full RAM image using WinPmem (Windows) or LiME kernel module (Linux) to preserve injected .NET assembly reflective load artifacts and in-memory RAT state; (2) active network connections via 'netstat -ano' (Windows) or 'ss -tunap' (Linux/macOS) to identify live C2 sessions to Tutamail infrastructure or unresolved IP endpoints spawned by node.exe/dotnet; (3) running process tree snapshot via 'Get-Process | Select-Object Id,Name,Path,Parent' (Windows) or 'ps auxf' (Linux/macOS) to document child processes spawned from npm.cmd or node.exe; (4) full copy of ~/.npm/_logs/, %APPDATA%\npm-cache\ (Windows), and the node_modules/.bin/ directory before any cache purge operation; (5) contents of package-lock.json and any yarn.lock files documenting resolved easy-day-js version and transitive dependency tree.

Step 2: Detection — Query endpoint logs for postinstall script execution originating from npm processes; look for child processes spawned by node.exe or npm.cmd making outbound HTTP/HTTPS connections (especially with TLS verification disabled). On Windows, inspect Defender for Endpoint alerts for reflective .NET injection (T1055.001) and new run key entries under HKCU\Software\Microsoft\Windows\CurrentVersion\Run. On macOS, check ~/Library/LaunchAgents and /Library/LaunchDaemons for unfamiliar plist entries created after June 17, 2026. On Linux, audit cron, systemd unit files, and ~/.bashrc or ~/.profile for injected persistence. Search npm audit logs and CI/CD pipeline logs for any reference to 'easy-day-js' as a resolved dependency. Enable or review Microsoft Defender XDR alerts for the reported detection signatures.

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: Deploy Sysmon with SwiftOnSecurity config to capture Event ID 1 (Process Create) filtered for node.exe/npm.cmd parent-child chains and Event ID 3 (Network Connection) for outbound connections from those processes. Query with: Get-WinEvent -LogName 'Microsoft-Windows-Sysmon/Operational' | Where-Object {$_.Id -eq 1 -and $_.Message -match 'npm|node'} | Select-Object TimeCreated, Message. On Linux/macOS, use osquery: SELECT name, path, cmdline, parent FROM processes WHERE parent IN (SELECT pid FROM processes WHERE name IN ('node','npm')) AND start_time > 1750118460; — timestamp corresponds to June 17, 2026 01:01 UTC. For postinstall hook detection, search CI/CD pipeline YAML artifacts and build logs for 'postinstall' script execution records referencing easy-day-js.

Evidence: BEFORE any system state changes, preserve: (1) Sysmon Event ID 1 (Process Create) logs showing npm.cmd spawning child processes during the postinstall hook phase of easy-day-js installation; (2) Windows Security Event Log Event ID 4688 (Process Creation) filtered for cmd.exe, powershell.exe, or dotnet.exe with parent process node.exe or npm.cmd; (3) HKCU\Software\Microsoft\Windows\CurrentVersion\Run registry hive snapshot exported via 'reg export HKCU\Software\Microsoft\Windows\CurrentVersion\Run run_keys_backup.reg'; (4) macOS Unified Log entries via 'log show --predicate "process == 'node'" --start 2026-06-17' capturing LaunchAgent plist creation events; (5) full export of ~/Library/LaunchAgents/ and /Library/LaunchDaemons/ with ls -la timestamps; (6) Linux systemd journal entries: 'journalctl --since 2026-06-17T01:01:00 -u "*" | grep -E "node|npm|easy-day"' and crontab -l output for all user accounts.

Step 3: Eradication — Remove all @mastra packages from affected environments and purge node_modules directories entirely; do not simply reinstall without version pinning to a verified clean release. Remove

'easy-day-js' and any transitive dependencies it introduced. Delete all identified persistence mechanisms: registry run keys (Windows), launch agent/daemon plists (macOS), and systemd/cron entries (Linux). Rotate all credentials, API keys, tokens, and SSH keys present on or accessible from compromised systems, including secrets stored in environment variables, .env files, and CI/CD secret stores (T1552.001 exposure). Revoke and reissue code-signing certificates if present on affected build systems.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST AC-2 (Account Management), NIST AC-3 (Access Enforcement), NIST AC-6 (Least Privilege), CIS 5.3 (Disable Dormant Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: For credential rotation without a PAM platform: enumerate all .env files, shell history, and CI/CD environment variable configs using 'grep -rE "(API_KEY|SECRET|TOKEN|PASSWORD|AWS_|GITHUB_TOKEN)" /project/ --include="*.env" --include="*.conf"' (Linux/macOS) or 'Get-ChildItem -Recurse -Filter *.env | Select-String -Pattern "KEY|SECRET|TOKEN"' (Windows). Export a baseline of SSH authorized_keys from all affected systems, revoke and replace key pairs. For GitHub Actions/GitLab CI, audit the repository Secrets UI and rotate every secret accessible by a runner that executed on a compromised host. Use 'npm cache verify' then 'npm cache clean --force' and validate node_modules removal with 'find . -name node_modules -type d -prune -exec rm -rf {} +' before reinstallation.

Evidence: BEFORE rotating credentials, revoking certificates, or deleting persistence artifacts: (1) capture a complete registry export of HKCU and HKLM Run/RunOnce keys from all affected Windows hosts; (2) binary-copy all LaunchAgent plist files from macOS ~/Library/LaunchAgents/ and /Library/LaunchDaemons/ — do not delete without copying, as plist contents may identify additional C2 endpoints not yet blocked; (3) collect all .env files and CI/CD secret environment variable exports as forensic copies before rotation to establish scope of credential exposure for breach notification assessment; (4) capture full node_modules/.bin/ contents and package-lock.json resolved dependency tree documenting every transitive package version pulled alongside easy-day-js; (5) export SSH known_hosts and authorized_keys from affected systems to document potential lateral movement staging.

Step 4: Recovery — Rebuild compromised developer workstations and CI/CD runners from trusted golden images rather than attempting in-place remediation, given the RAT's persistence capability. Re-establish npm dependencies from a clean lockfile with all packages pinned to verified versions and validated against known-good checksums. Re-run builds in an isolated environment and validate build artifact integrity before promoting to production. Monitor rebuilt systems for 30 days using EDR telemetry for signs of reinfection or residual C2 beaconing. Confirm Microsoft Defender detections are resolving clean on rebuilt systems.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST CM-4 — no mapped control in knowledge base for image-based recovery; see CIS 4.6, CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Validate golden image integrity before reimaging using SHA-256 checksums of the image file compared against a pre-incident reference stored offline. After rebuild, verify package-lock.json integrity using 'npm ci' (which enforces exact lockfile versions) rather than 'npm install'. Validate npm package checksums post-install: 'npm audit signatures' (npm CLI v8.7+) verifies registry signature for each installed package. For CI/CD artifact validation without enterprise tooling, generate and compare SHA-256 hashes of build output binaries before and after: 'sha256sum dist/** > build.sha256' and store the reference hash in a Git-signed commit. For 30-day monitoring without EDR, configure Sysmon to alert on any process establishing outbound connections to port 443 from node.exe or dotnet.exe.

Evidence: BEFORE reimaging or rebuilding any system: (1) acquire a full forensic disk image using FTK Imager or 'dd if=/dev/sda bs=512 conv=noerror,sync | gzip > host_image.gz' — the RAT's persistence mechanism may leave artifacts in slack space or alternate data streams not visible at the OS level; (2) capture any remaining volatile memory not yet imaged from Step 1; (3) preserve the complete CI/CD pipeline execution logs documenting exactly which build jobs ran on compromised runners and which artifact outputs they produced — these artifacts may themselves be

poisoned and must be quarantined before the logs are overwritten by new pipeline runs; (4) export EDR/Defender telemetry covering the June 17, 2026 UTC window through isolation time for each host as a dated forensic record before agent reinstallation on the new image.

Step 5: Post-Incident — Conduct a control gap review against NIST SI-4 (system monitoring), NIST CM-7 (least functionality for build systems), and CIS 2.1/2.3 (software inventory and unauthorized software removal). Implement npm package integrity verification using lockfile pinning and Subresource Integrity or equivalent checksum validation (addresses CWE-494, no mapped control for npm-specific supply chain integrity in the knowledge base beyond CIS 7.1/7.2 vulnerability management process). Evaluate adoption of a private npm registry proxy with allowlisting to prevent direct public registry pulls in CI/CD. Require MFA on all npm maintainer accounts your organization controls (CIS 6.5). Establish a process for monitoring npm account takeover advisories through the npm security advisories feed.

NIST Phase: Post Incident

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

Controls: NIST SI-2 (Flaw Remediation), NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 2.3 (Address Unauthorized Software), CIS 6.5 (Require MFA for Administrative Access), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Implement a free Verdaccio private npm registry proxy (verdaccio.org) in CI/CD to intercept and allowlist approved @mastra package versions, preventing direct public registry pulls. Configure .npmrc in all project repos with 'registry=https://your-verdaccio-instance' and 'audit=true'. For npm account takeover monitoring without a commercial threat intel feed, subscribe to the npm security advisories RSS feed at https://github.com/nicolo-ribaudo/nicolo-ribaudo/security/advisories and configure a GitHub Actions workflow to poll the npm security endpoint 'npm audit --json' on a daily cron schedule. For software inventory of npm packages, run 'npm list --all --json > npm_inventory_$(date +%Y%m%d).json' on all build systems weekly and diff against the previous week's output to detect unauthorized package additions.

Evidence: For the post-incident lessons-learned review, assemble: (1) complete timeline of npm install/update command execution across all affected systems derived from npm debug logs, CI/CD pipeline timestamps, and endpoint process creation logs — this establishes dwell time and blast radius for breach notification scope; (2) inventory of all credentials, API keys, tokens, and certificates confirmed exposed during eradication, mapped to owning systems and downstream services — required for breach notification threshold assessment; (3) a diff of pre- and post-incident package-lock.json files across all affected repositories to document the full transitive dependency tree introduced by easy-day-js; (4) CI/CD artifact integrity report identifying which build outputs were produced on compromised runners and therefore must be treated as potentially poisoned and withheld from production.

Detection Guidance

Primary indicators: (1) Presence of 'easy-day-js' in any node_modules directory, package-lock.json, or npm audit output. (2) Outbound network connections from node.exe, npm, or npm.cmd to non-CDN external IPs during or after 'npm install' execution, particularly connections that bypass TLS validation. (3) On Windows: new entries under HKCU\Software\Microsoft\Windows\CurrentVersion\Run or HKLM\...\Run created by node or npm processes (T1547.001); .NET assembly loaded reflectively into a non-.NET host process (T1055.001); PowerShell execution (T1059.001) spawned as a child of npm. (4) On macOS: new .plist files in ~/Library/LaunchAgents or /Library/LaunchDaemons with creation timestamps after June 17, 2026, associated with Node.js paths (T1543.001). (5) On Linux: new cron entries, systemd unit files, or modifications to shell profile files (.bashrc, .profile, .zshrc) referencing npm cache paths or temporary directories (T1543.004, T1546). (6) Cryptocurrency wallet file access (T1552.001), file system events touching wallet.dat or browser extension storage paths for Metamask, Exodus, or similar wallets. (7) Encoded or obfuscated JavaScript execution in

postinstall context (T1027, T1059.007). Microsoft Defender Antivirus, Defender for Endpoint, and Defender XDR have reported active detection coverage; review these alert queues for any triggered signatures related to this campaign. Log sources to query: EDR process creation logs (parent-child: npm → cmd/sh → curl/powershell/dotnet), Windows Event ID 4688 (process creation with command line), macOS Unified Log (launchd activity), Linux auditd (execve calls from npm postinstall context), CI/CD pipeline build logs for dependency resolution including easy-day-js.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`tutamail[.]com`	Email domain used by attacker accounts: ehindero2016@tutamail[.]com and sergey2016@tutamail[.]com — associated with the compromised npm maintainer identity	HIGH
DOMAIN	`easy-day-js`	Typosquatted npm package name used as the malicious dependency injected into 140+ @mastra packages; presence in node_modules or lockfile indicates exposure	HIGH

Framework Mappings

MITRE-ATTACK

- **T1055.001** — Dynamic-link Library Injection
- **T1071.001** — Web Protocols
- **T1552.001** — Credentials In Files
- **T1059.001** — PowerShell
- **T1041** — Exfiltration Over C2 Channel
- **T1055** — Process Injection
- **T1014** — Rootkit
- **T1059.007** — JavaScript
- **T1546.012** — Image File Execution Options Injection
- **T1543.004** — Launch Daemon
- **T1078** — Valid Accounts
- **T1547.001** — Registry Run Keys / Startup Folder
- **T1543.001** — Launch Agent
- **T1554** — Compromise Host Software Binary
- **T1552** — Unsecured Credentials
- **T1562.001** — Disable or Modify Tools
- **T1195.001** — Compromise Software Dependencies and Development Tools

- **T1027** — Obfuscated Files or Information
- **T1546** — Event Triggered Execution
- **T1078.004** — Cloud Accounts

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **AC-6** — Least Privilege
- **AC-2** — Account Management
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **AC-3** — Access Enforcement
- **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

- **6.1** — Establish an Access Granting Process
- **6.2** — Establish an Access Revoking Process
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **3.10** — Encrypt Sensitive Data in Transit
- **15.1** — Establish and Maintain an Inventory of Service Providers
- **8.2** — Collect Audit Logs

SOC2-TSC

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets
- **CC9.2** — Manages risks associated with vendors and business partners

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control
- **164.312(e)(1)** — Transmission Security

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.21** — Managing information security in the ICT supply chain

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program
- **DE.CM-01** — Networks and network services are monitored

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1055.001	Dynamic-link Library Injection	Defense-Evasion
T1071.001	Web Protocols	Command-And-Control
T1552.001	Credentials In Files	Credential-Access
T1059.001	PowerShell	Execution
T1041	Exfiltration Over C2 Channel	Exfiltration
T1055	Process Injection	Defense-Evasion
T1014	Rootkit	Defense-Evasion
T1059.007	JavaScript	Execution
T1546.012	Image File Execution Options Injection	Privilege-Escalation
T1543.004	Launch Daemon	Persistence
T1078	Valid Accounts	Defense-Evasion
T1547.001	Registry Run Keys / Startup Folder	Persistence
T1543.001	Launch Agent	Persistence
T1554	Compromise Host Software Binary	Persistence
T1552	Unsecured Credentials	Credential-Access
T1562.001	Disable or Modify Tools	Defense-Evasion
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access

Technique ID	Technique Name	Tactic
T1027	Obfuscated Files or Information	Defense-Evasion
T1546	Event Triggered Execution	Privilege-Escalation
T1078.004	Cloud Accounts	Defense-Evasion

Sources

Source	URL	Tier
Microsoft Security Blog	https://www.microsoft.com/en-us/security/blog/2026/06/17/postinstal...	T1
	https://www.microsoft.com/en-us/security/blog/2026/06/17/postinstal...	T1
	https://thehackernews.com/2026/06/144-mastra-npm-packages-compromis ...	T3
	https://securityboulevard.com/2026/06/easy-day-js-targets-mastra-de...	T3
140+ Packages Backdoored via easy-day-js Typosquat - StepSecurity	https://www.stepsecurity.io/blog/mastra-npm-packages-compromised-us...	T3

Generated 2026-06-18 07:12 UTC by TJS Security Command Center