INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-06 08:41 UTC

Axios Prototype Pollution Gadgets: Response Tampering, Data Exfiltration, and Request Hijacking (CVE-2026-42033)

CVE VULNERABILITY | HIGH | CVSS 8.1

SCC Item ID	SCC-CVE-2026-0127
Type	CVE Vulnerability
CVE ID	CVE-2026-42033
Severity	HIGH
CVSS Base Score	8.1
EPSS Score	0.0010 (28th percentile)
Affected Products	axios (npm), versions including axios-1.15.0 confirmed affected; full version range unconfirmed pending NVD full publication
Published	2026-05-05T00:26:29Z
Discovery Source	Osv

Executive Summary

A high-severity vulnerability (CVE-2026-42033) in the axios HTTP client library, widely used across JavaScript and Node.js applications, enables attackers to manipulate HTTP responses, intercept outbound requests, and exfiltrate data through prototype pollution gadget chains. Organizations running web applications or backend services that depend on axios are potentially exposed, with confirmed impact in axios version 1.15.0. The business risk centers on unauthorized data extraction and request tampering in any application that processes attacker-influenced input through axios.

Technical Analysis

CVE-2026-42033 identifies exploitable prototype pollution gadget chains in the axios npm library (CWE-94, CWE-1321). Prototype pollution occurs when attacker-controlled input reaches Object.prototype, altering inherited properties across all JavaScript objects in the runtime. In axios, these gadgets enable three attack classes: (1) response tampering, modifying HTTP response data before application logic processes it; (2) data exfiltration, redirecting or copying request/response content to attacker-controlled destinations (maps to T1048: Exfiltration Over Alternative Protocol); (3) request hijacking, intercepting or modifying outbound HTTP requests (maps to T1565.002: Transmitted Data Manipulation and T1190: Exploit Public-Facing Application). Confirmed affected version: axios-1.15.0 (per opensearch-project/OpenSearch-Dashboards issue #11848 and OSV

advisory GHSA-pf86-5x62-jrwf). Full affected version range and CVSS vector are unconfirmed pending complete NVD publication. CVSS base score reported as 8.1 (High); EPSS score 0.00103 (27.7th percentile, per FIRST EPSS specification) indicates low current exploitation activity. No CISA KEV listing as of analysis date. Patched version not independently verified, monitor GHSA-pf86-5x62-jrwf and NVD for update.

Action Checklist

1. Containment. Inventory all applications and services using axios as a direct or transitive npm dependency. Flag any instance running axios-1.15.0 or versions in the unconfirmed affected range. Prioritize internet-facing Node.js services that process user-supplied or third-party input through axios request/response cycles. Implement input validation or JSON schema verification on untrusted input before passing to axios request/response handlers as a temporary mitigation where feasible.
2. Detection. Query your software composition analysis (SCA) tool or run 'npm list axios' across repositories and deployed environments to identify affected versions. Review application logs for anomalous outbound HTTP requests (unexpected destinations, unusual payloads, response content deviations). In SIEM, alert on unexpected external connections from Node.js processes (correlate with T1048 exfiltration patterns). Check OSV advisory GHSA-pf86-5x62-jrwf for any published IOCs or payload signatures as NVD publication completes.
3. Eradication. Upgrade axios to the patched version once confirmed by the GHSA-pf86-5x62-jrwf advisory or NVD record. Do not upgrade blindly to latest without verifying the fix commit addresses prototype pollution gadget chains. Validate transitive dependencies: packages that depend on axios may pull in the affected version independently. Run 'npm audit' post-upgrade to confirm resolution. For projects where upgrade is not immediately possible, evaluate whether JSON schema validation or input sanitization can block attacker-controlled prototype-polluting input before it reaches axios.
4. Recovery. After patching, re-run SCA scans to confirm axios version resolution across all dependency trees. Validate application HTTP behavior against expected baselines, confirm response integrity, outbound request destinations, and header content are unchanged. Monitor application telemetry for 72 hours post-patch for residual anomalies. Update your software bill of materials (SBOM) to reflect the remediated version.
5. Post-Incident. Review dependency management policies: automated dependency update tooling (Dependabot, Renovate) should be configured to alert on high-severity advisories for transitive dependencies, not just direct ones. Assess whether prototype pollution sinks in your application code create additional gadget exposure beyond axios. Add CWE-94 and CWE-1321 to your secure code review checklist and SCA rule sets. Track GHSA-pf86-5x62-jrwf for additional technical disclosure as NVD publication finalizes.

IR / Forensic Enrichment

Triage Priority	URGENT

Escalation Criteria	Escalate to CISO and legal/privacy counsel immediately if application logs confirm any successful prototype pollution exploitation (evidence of response tampering, modified outbound request destinations, or unexpected data in axios response transformers) on services that handle PII, PHI, or financial data, as this constitutes a potential data exfiltration event triggering GDPR 72-hour notification, HIPAA breach assessment, or PCI DSS incident reporting obligations.
Recovery Notes	Post-patch, verify that Object.prototype is clean in running Node.js processes by querying 'require("assert").strictEqual(Object.prototype.baseURL, undefined)' and similar checks for axios config keys ('headers', 'transformResponse', 'transformRequest', 'baseURL') — a successful pre-patch exploitation of CVE-2026-42033 may have left persistent prototype mutations that survive the library upgrade if the process was not restarted. Restart all Node.js services after patching rather than relying on hot-reload, as prototype pollution effects persist in the V8 heap for the lifetime of the process. Monitor outbound HTTP destinations and response Content-Type headers for 72 hours post-restart, specifically watching for any axios requests routing to unexpected hosts or returning unexpected content that would indicate a gadget chain effect lingering from a prior attack.
Forensic Artifacts	Node.js process heap state at time of suspected exploitation — capture via 'kill -USR2 ' (requires heapdump module) or v8 inspector heap snapshot; look for Object.prototype properties matching axios config keys ('baseURL', 'headers', 'transformResponse') which would confirm prototype pollution gadget chain execution for CVE-2026-42033 | Web application / reverse proxy access logs (nginx access.log, Apache access.log, or AWS ALB access logs) — search for inbound request bodies or query parameters containing '__proto__', 'constructor', 'prototype' key patterns that were the attack vector delivering the pollution payload to the axios-consuming service | Outbound network flow logs or proxy logs (Squid, mitmproxy capture, or cloud VPC flow logs) from Node.js service hosts — filter on connections from node process PIDs to destinations not in your application's known API/CDN whitelist; CVE-2026-42033 request-hijacking gadgets would redirect axios outbound calls to attacker-controlled infrastructure | package-lock.json and node_modules/axios/package.json from all deployed environments at time of incident — these establish the exact axios version (1.15.0 confirmed affected) and the full transitive dependency tree that resolved to the vulnerable version, critical for scope assessment | Application-level axios interceptor logs or HTTP response body logs if instrumented — response tampering gadget chains from CVE-2026-42033 would manifest as modified response data that diverges from upstream API responses; compare against upstream source-of-truth responses for the same endpoints to identify tampering artifacts

Per-Action IR Details

Containment — Inventory all applications and services using axios as a direct or transitive npm dependency. Flag any instance running axios-1.15.0 or versions in the unconfirmed affected range. Prioritize internet-facing Node.js services that process user-supplied or third-party input through axios request/response cycles. Restrict untrusted input paths to axios as a temporary measure where feasible.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST CM-8 (System Component Inventory), NIST SI-2 (Flaw Remediation), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 2.1 (Establish and Maintain a Software Inventory)

Compensating: Run 'npm list axios --all 2>/dev/null | grep axios' recursively across all Node.js project roots to enumerate direct and transitive axios versions without SCA tooling. Pipe output to a file for audit trail: 'find /opt /srv /home -name package.json -not -path "*/node_modules/*" -exec dirname {} \; | xargs -I{} sh -c "cd {} && npm list axios

--all 2>/dev/null | grep axios && echo [PATH: {}]" > axios_inventory_$(date +%Y%m%d).txt'. For immediate input restriction on Express-based services, add a middleware that rejects requests with '__proto__', 'constructor', or 'prototype' keys in JSON body before they reach axios call sites.

Evidence: Before restricting input paths, capture: (1) current Node.js process list with full command-line arguments ('ps aux | grep node' on Linux; 'Get-Process node | Select-Object Id,Path,CommandLine' on Windows) to establish a baseline of running axios-consuming services; (2) snapshot of all package-lock.json and yarn.lock files showing resolved axios versions — these are the authoritative record of what version was deployed at time of incident; (3) network connection state of Node.js processes ('ss -tnp | grep node' or 'netstat -anp | grep node') to identify any already-established anomalous outbound connections that prototype pollution gadget chains may have initiated for data exfiltration.

Detection — Query your software composition analysis (SCA) tool or run 'npm list axios' across repositories and deployed environments to identify affected versions. Review application logs for anomalous outbound HTTP requests (unexpected destinations, unusual payloads, response content deviations). In SIEM, alert on unexpected external connections from Node.js processes (correlate with T1048 exfiltration patterns). Check OSV advisory GHSA-pf86-5x62-jrwf for any published IOCs or payload signatures as NVD publication completes.

NIST Phase: Detection Analysis

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

Controls: NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-12 (Audit Record Generation), NIST IR-5 (Incident Monitoring), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Deploy Sysmon with a configuration that captures Event ID 3 (Network Connection) for node.exe processes — filter on DestinationPort not in [80,443] or DestinationIP not in your known-good CDN/API ranges to surface prototype-pollution-driven exfiltration. For application log review without SIEM, use: 'grep -E "(proto__|constructor|\[object Object\])" /var/log/nginx/access.log' to find pollution payloads hitting your app, and 'grep -Ev "(api\.yourdomain\.com|known-cdn\.com)" node_app.log | grep -i "http://\|https://"' to find unexpected outbound axios destinations logged by your application. Write a Sigma rule targeting Sysmon EID 3 for node.exe spawning connections to non-whitelisted destinations and run it against collected Sysmon EVTX files using sigma-cli with the EVTX backend.

Evidence: Before alerting/blocking, preserve: (1) full application access logs from the Node.js service (e.g., Express morgan logs or PM2 output logs at ~/.pm2/logs/) showing inbound request payloads — prototype pollution attacks for CVE-2026-42033 would manifest as JSON bodies or query parameters containing '__proto__', 'constructor.prototype', or nested object keys targeting axios config properties like 'headers', 'baseURL', or 'transformResponse'; (2) outbound HTTP request logs from axios interceptors if instrumented, or from a network proxy/WAF, showing unexpected destination URLs or modified request headers that a request-hijacking gadget chain would produce; (3) Node.js heap dump if any process shows anomalous behavior ('node --inspect' + Chrome DevTools heap snapshot, or 'kill -USR2 ' if using heapdump module) to capture polluted Object.prototype state at time of suspected exploitation (MITRE ATT&CK T1048 — Exfiltration Over Alternative Protocol).

Eradication — Upgrade axios to the patched version once confirmed by the GHSA-pf86-5x62-jrwf advisory or NVD record. Do not upgrade blindly to latest without verifying the fix commit addresses prototype pollution gadget chains. Validate transitive dependencies: packages that depend on axios may pull in the affected version independently. Run 'npm audit' post-upgrade to confirm resolution. For projects where upgrade is not immediately possible, evaluate whether JSON schema validation or input sanitization can block attacker-controlled prototype-polluting input before it reaches axios.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST CM-3 (Configuration Change Control), NIST SA-10 (Developer Configuration Management), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: For teams that cannot immediately upgrade axios, implement prototype pollution hardening at the Node.js runtime level: add 'Object.freeze(Object.prototype)' early in your application entrypoint (before any require() calls that load axios) to prevent gadget chain execution — note this may break libraries that legitimately extend Object.prototype, so test in staging first. Alternatively, use the 'freeze-prototype' npm package for a drop-in guard. For input sanitization as a compensating control, integrate 'defu' sanitizer or write a recursive function that strips '__proto__', 'constructor', and 'prototype' keys from all JSON.parse() output before it flows into axios config or request parameters. Verify the fix commit in the GHSA-pf86-5x62-jrwf advisory specifically addresses gadget chains in axios response transformers and/or config merging logic before approving the upgrade.

Evidence: Before upgrading, preserve: (1) a full snapshot of node_modules/axios/lib/ directory — specifically 'core/mergeConfig.js', 'utils.js', and 'defaults/index.js' which are the typical locations of prototype pollution sinks in axios — this establishes the vulnerable code state for post-incident analysis; (2) output of 'npm list axios --all' showing the full dependency tree resolving to axios-1.15.0, capturing which transitive dependents (e.g., other npm packages in your stack) pulled in the vulnerable version; (3) 'npm audit --json > audit_pre_patch_$(date +%Y%m%d).json' to document the pre-patch vulnerability state as an artifact of record per NIST AU-11 (Audit Record Retention) requirements.

Recovery — After patching, re-run SCA scans to confirm axios version resolution across all dependency trees. Validate application HTTP behavior against expected baselines — confirm response integrity, outbound request destinations, and header content are unchanged. Monitor application telemetry for 72 hours post-patch for residual anomalies. Update your software bill of materials (SBOM) to reflect the remediated version.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST SI-7 (Software, Firmware, and Information Integrity), NIST CP-10 (System Recovery and Reconstitution), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST CM-8 (System Component Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 2.1 (Establish and Maintain a Software Inventory)

Compensating: Validate HTTP response integrity post-patch using a simple curl-based regression test: capture a baseline response body hash for key API endpoints ('curl -s https://yourapp/api/endpoint | sha256sum') pre-patch, then re-run post-patch and compare — response tampering via prototype pollution gadget chains would have altered these hashes during exploitation. For SBOM update without enterprise tooling, generate a CycloneDX SBOM using 'npx @cyclonedx/cyclonedx-npm --output-file sbom_$(date +%Y%m%d).json' and store it in your version control system tagged to the patched release commit. Monitor for 72 hours using 'tail -f' on application logs piped through 'grep -E "(proto__|unexpected_host|Content-Type mismatch)"' as a low-cost residual anomaly detector.

Evidence: Before closing recovery: (1) 'npm list axios --all' output post-patch confirming zero instances of axios-1.15.0 remain in any dependency tree — this is your primary recovery verification artifact; (2) comparison of outbound HTTP request destinations logged before and after patching to confirm no prototype-pollution-induced baseURL or header overrides persisted (gadget chains in axios can set persistent config mutations that survive the initial payload if Object.prototype was polluted before the patch); (3) application response payload samples from key endpoints pre- and post-patch to verify response integrity was not altered by a prior tampering gadget chain that may have cached malicious transformResponse functions.

Post-Incident — Review dependency management policies: automated dependency update tooling (Dependabot, Renovate) should be configured to alert on high-severity advisories for transitive dependencies, not just direct ones. Assess whether prototype pollution sinks in your application code create additional gadget exposure beyond axios. Add CWE-1321 to your secure code review checklist and SCA rule sets. Track GHSA-pf86-5x62-jrwf for additional technical disclosure as NVD publication finalizes.

NIST Phase: Post Incident

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

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SA-11 (Developer Testing and Evaluation), NIST SI-2 (Flaw Remediation), NIST RA-3 (Risk Assessment), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For SCA rule sets without enterprise tooling, add a custom .nsprc or 'npm audit' policy file that blocks any advisory tagged CWE-1321 from passing CI/CD pipelines. Configure Dependabot via '.github/dependabot.yml' with 'target-branch' and 'open-pull-requests-limit' settings that force transitive dependency updates — add a 'groups' block specifically for HTTP client libraries including axios to bundle updates. For prototype pollution sink detection in application code, run 'npx is-mergeable-object' or use semgrep with the 'javascript.lang.security.audit.prototype-pollution' ruleset ('semgrep --config=p/javascript ci') against your codebase to enumerate additional CWE-1321 sinks beyond axios that could be exploited via similar gadget chains.

Evidence: For lessons-learned documentation, compile: (1) timeline artifact showing the delta between GHSA-pf86-5x62-jrwf publication date and your organization's detection/remediation date — this gap is your key metric for improving transitive dependency monitoring maturity per NIST IR-8 (Incident Response Plan) update requirements; (2) list of all application code locations that call axios with externally-controlled config parameters (baseURL, headers, transformResponse, transformRequest) identified via 'grep -rn "axios(\|axios\.get\|axios\.post\|axios\.create" src/' — these are your residual gadget chain attack surface; (3) Dependabot/Renovate alert configuration diff showing before/after state of transitive dependency coverage, documenting the control improvement.

Detection Guidance

Run 'npm list axios --all' in each repository and 'npm audit' to surface axios-1.15.0 or other affected versions in direct and transitive dependency trees. In deployed environments, use SCA tooling (Snyk, Dependabot, OWASP Dependency-Check) to scan running artifact inventories. Behavioral detection: monitor Node.js process network activity for unexpected outbound connections, particularly to external IPs not in your application's known destination list (SIEM correlation on T1048). Alert on HTTP response anomalies, unexpected content-type changes, appended or modified response bodies, or responses redirected to unrecognized endpoints. Review OSV advisory GHSA-pf86-5x62-jrwf as NVD publication completes for any confirmed payload signatures or proof-of-concept indicators. No public IOCs (IPs, domains, hashes) are confirmed at analysis time.

Framework Mappings

MITRE-ATTACK

- **T1048** — Exfiltration Over Alternative Protocol
- **T1565.002** — Transmitted Data Manipulation
- **T1190** — Exploit Public-Facing Application

NIST-800-53R5

- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **SI-4** — System Monitoring

CIS-V8

- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

- **8.2** — Collect Audit Logs

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.34** — Privacy and protection of personal information

HIPAA-SECURITY

- **164.308(a)(6)(ii)** — Response and Reporting

SOC2-TSC

- **CC7.4** — Responds to identified security incidents

NIST-CSF-2

- **DE.CM-01** — Networks and network services are monitored

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1048	Exfiltration Over Alternative Protocol	Exfiltration
T1565.002	Transmitted Data Manipulation	Impact
T1190	Exploit Public-Facing Application	Initial-Access

Sources

Source	URL	Tier
osv	https://osv.dev/vulnerability/GHSA-pf86-5x62-jrwf	T3
CVE-2026-42033 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-42033	T1
Linux Distros Unpatched Vulnerability : CVE-2026-42033 | Tenable®	https://www.tenable.com/plugins/nessus/310515	T3
CVE-2026-42033 - HIGH PRIORITY Vulnerability | SOC Defenders	https://socdefenders.ai/cves/CVE-2026-42033	T3
CVE-2026-42033 (High) detected in axios-1.15.0.tgz #11848 - GitHub	https://github.com/opensearch-project/OpenSearch-Dashboards/issues/...	T3

DISCLAIMER

Generated 2026-05-06 08:41 UTC by TJS Security Command Center