INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-05 06:57 UTC

GHSA-j5f8-grm9-p9fc: Axios: Proxy-Authorization header leaks to redirect target when proxy is re-eval

CVE VULNERABILITY | HIGH | CVSS 7.5

SCC Item ID	SCC-CVE-2026-0266
Type	CVE Vulnerability
CVE ID	CVE-2026-44486
Severity	HIGH
CVSS Base Score	7.5
Affected Products	axios (npm), specific affected versions not confirmed from available sources
Published	2026-06-04T14:15:01Z
Discovery Source	Osv

Executive Summary

A credential leakage vulnerability in the Axios HTTP client library (CVE-2026-44486) causes Proxy-Authorization headers to be forwarded to redirect destinations when proxy configuration changes state mid-request. Any application using Axios to make proxied HTTP requests is potentially exposing proxy credentials to unintended third-party servers. Organizations relying on Axios-based services for API integrations, backend communication, or supply chain tooling face credential compromise and unauthorized access risk.

Technical Analysis

CVE-2026-44486 affects the Axios npm library for Node.js. The flaw is in redirect handling logic: when a proxy is re-evaluated as a direct connection during a redirect sequence, Axios fails to strip the Proxy-Authorization header before forwarding the request to the redirect target. This constitutes improper header sanitization (CWE-200: Exposure of Sensitive Information to an Unauthorized Actor; CWE-522: Insufficiently Protected Credentials). An attacker controlling or observing the redirect target can harvest proxy credentials passively, requiring no active exploitation beyond inducing a redirect. MITRE ATT&CK techniques T1071.001 (Application Layer Protocol: Web Protocols) and T1552 (Unsecured Credentials) apply. As of publication date, specific affected versions and patch status have not been confirmed in NVD. Monitor NVD (https://nvd.nist.gov/vuln/detail/CVE-2026-44486) and GHSA-j5f8-grm9-p9fc for updates. CVSS base score: 7.5 (High). EPSS data not yet populated. Not listed on CISA KEV as of configuration date.

Action Checklist

1. Step 1: Containment. Audit all production Node.js applications and build pipelines for direct or transitive dependencies on the Axios npm package. Where Axios is confirmed present and proxy authentication is in use, temporarily strip Proxy-Authorization headers from outbound requests or route affected services through an authenticated gateway until a patched version is available. Reference: NIST AC-4 (Information Flow Enforcement), enforce approved authorizations for information flowing between connected systems.
2. Step 2: Detection. Query application logs and proxy access logs for HTTP redirect chains (HTTP 301/302/307/308 responses) where the subsequent request to the redirect destination carries an Authorization or Proxy-Authorization header value. Search SIEM for outbound requests where the destination host differs from the original proxy target but the Authorization header is present. Enable verbose header logging in staging/test environments to confirm exposure. Reference: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting); CIS 8.2 (Collect Audit Logs).
3. Step 3: Eradication. Upgrade Axios to a patched version once one is published and confirmed by the maintainers (monitor GHSA-j5f8-grm9-p9fc and the official Axios GitHub release page for patch availability). If a patched version is not yet available, implement middleware or a wrapper that explicitly removes Proxy-Authorization and Authorization headers from any request following a redirect to a host outside the original proxy target. Change proxy account credentials on all affected systems. Reference: CIS 7.4 (Perform Automated Application Patch Management); NIST IA-4 (Identifier Management).
4. Step 4: Recovery. After patching, re-run dependency scans across all affected repositories to confirm the vulnerable Axios version is no longer present in any dependency tree, including transitive dependencies. Validate proxy authentication flows in staging with redirect scenarios enabled. Monitor proxy and application logs for 30 days post-remediation for anomalous authentication patterns. Reference: NIST AU-6 (Audit Record Review, Analysis, and Reporting).
5. Step 5: Post-Incident. Conduct a dependency audit to identify all third-party libraries handling HTTP headers in proxied contexts. Implement automated software composition analysis (SCA) in CI/CD pipelines to flag vulnerable library versions before deployment. Review header sanitization requirements in secure coding standards. Reference: NIST SI-4 (System Monitoring), apply as preventive system monitoring control; CIS 2.1 (Establish and Maintain a Software Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process).

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to CISO and legal/privacy counsel if proxy access logs confirm that Proxy-Authorization headers were forwarded to any external third-party host during the dwell period, as this constitutes credential exposure to an uncontrolled party and may trigger breach notification obligations under GDPR, CCPA, or contractual SLAs depending on the nature of services authenticated by the leaked proxy credentials.

Recovery Notes	After patching Axios and rotating proxy credentials, verify recovery by running a controlled redirect simulation in staging — issue an HTTP request through the patched Axios instance to a host that returns a 302 to a different FQDN and confirm via packet capture that the follow-up request contains no Authorization or Proxy-Authorization header. Monitor all forward-proxy access logs for 30 days post-rotation for any authentication attempt using the pre-rotation credential (base64-encoded Basic auth string), which would indicate the leaked credential is being actively replayed by a third party. Retain proxy logs covering the full vulnerability dwell period in cold storage for a minimum of 12 months to support any downstream forensic or compliance review.
Forensic Artifacts	Forward proxy access logs (Squid, Nginx, corporate proxy) with full request header logging enabled — specifically entries showing HTTP 30x responses followed by a subsequent request to the Location header destination where the 'Authorization' or 'Proxy-Authorization' header is present, which is the direct artifact of CVE-2026-44486 exploitation | node_modules/axios/package.json and package-lock.json from each production application server — these establish the exact vulnerable Axios version deployed and the resolution chain that introduced it, including whether it arrived as a direct or transitive dependency | Network packet captures (pcap) of the Node.js application's outbound TCP connections on port 80/443 during periods when proxied HTTP requests with redirects occurred — the pcap will show the raw HTTP headers in the follow-up GET to the redirect destination, providing definitive proof of credential leakage | Environment variable state at time of incident (redacted credential values, preserving variable names and proxy host configuration) — documents which proxy endpoints and credential identities were configured in Axios at the time of exposure, establishing the scope of what was leaked | CI/CD pipeline build logs showing historical npm install or npm ci output for all affected repositories — these establish the timeline of when the vulnerable Axios version was first introduced into each build artifact, enabling accurate dwell-time calculation for breach notification assessment

Per-Action IR Details

Step 1: Containment — Audit all production Node.js applications and build pipelines for direct or transitive dependencies on the Axios npm package. Where Axios is confirmed present and proxy authentication is in use, temporarily disable proxy credential passthrough or route affected services through an authenticated gateway that strips outbound headers until a patched version is available. Reference: NIST AC-4 (Information Flow Enforcement) — enforce approved authorizations for information flowing between connected systems.

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), NIST AC-6 (Least Privilege), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory)

Compensating: Run 'npm ls axios --all' in each production repo root to enumerate direct and transitive Axios presence; for monorepos use 'find . -name package-lock.json | xargs grep -l axios' to locate affected lockfiles. For immediate header stripping without a SIEM, deploy a lightweight Node.js proxy middleware using http-proxy-middleware configured to delete the 'Proxy-Authorization' and 'Authorization' headers on any response with a Location header (HTTP 301/302/307/308) before the redirect is followed. Two-person team can script this as a pre-commit hook and a runtime patch applied via NODE_OPTIONS=--require ./strip-auth-redirect-patch.js.

Evidence: Before disabling proxy credential passthrough, capture: (1) the current Axios version from each affected application's node_modules/axios/package.json and the resolved version in package-lock.json to establish scope; (2) outbound proxy access logs showing the original proxy target hostname and any Location header destinations from the 24–72 hours prior to containment, to determine whether redirect chains to third-party hosts already occurred; (3) environment variable dumps (scrubbed of secrets) confirming whether HTTPS_PROXY or HTTP_PROXY env vars carry credential-bearing URLs that were passed to Axios config.

Step 2: Detection — Query application logs and proxy access logs for HTTP redirect chains (HTTP 301/302/307/308 responses) where the subsequent request to the redirect destination carries an Authorization or Proxy-Authorization header value. Search SIEM for outbound requests where the destination host differs from the original proxy target but the Authorization header is present. Enable verbose header logging in staging/test environments to confirm exposure. Reference: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting); CIS 8.2 (Collect Audit Logs).

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-3 (Content Of Audit Records), CIS 8.2 (Collect Audit Logs)

Compensating: Without a SIEM, use mitmproxy or Wireshark on the Node.js host's outbound interface filtered to 'http.request.method == GET || http.request.method == POST' with display filter 'http.authorization' to capture any request carrying an Authorization header to a non-proxy destination. For log-based detection, run: 'awk "/30[12378]/{found=1} found && /Authorization/{print; found=0}" /var/log/squid/access.log' (adjust path for your proxy daemon) to surface redirect-followed requests containing auth headers. Deploy the Sigma rule targeting HTTP client credential leakage on redirect if your log pipeline supports it; the rule should correlate a 30x response from proxy_host with a subsequent request to redirect_destination where headers contain 'Proxy-Authorization'.

Evidence: Capture before analysis: (1) Squid, Nginx, or corporate forward-proxy access logs showing CONNECT and GET/POST entries with full request headers enabled — specifically look for entries where 'X-Forwarded-For' or 'Via' identifies the Node.js application as origin and the destination FQDN does not match the configured proxy endpoint; (2) application-level stdout/stderr logs from the Node.js process for Axios 'maxRedirects' behavior and any unhandled redirect events; (3) network packet capture (pcap) of the application's outbound TCP connections on port 80/443 around the time of any 301/302/307/308 response, preserving HTTP headers to confirm whether 'Proxy-Authorization' appears in the follow-up GET to the redirect target.

Step 3: Eradication — Upgrade Axios to a patched version once one is published and confirmed by the maintainers (monitor GHSA-j5f8-grm9-p9fc and the official Axios GitHub release page for patch availability). If a patched version is not yet available, implement middleware or a wrapper that explicitly removes Proxy-Authorization and Authorization headers from any request following a redirect to a host outside the original proxy target. Rotate all proxy credentials that may have been exposed. Reference: CIS 7.4 (Perform Automated Application Patch Management); D3-CRO (Credential Rotation).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: CIS 7.4 (Perform Automated Application Patch Management), CIS 7.2 (Establish and Maintain a Remediation Process), NIST AC-2 (Account Management)

Compensating: For teams without automated patch tooling: pin the patched Axios version explicitly in package.json using an exact version string (no ^ or ~ range operators) and run 'npm ci' rather than 'npm install' in all CI pipelines to enforce the lockfile. For the credential rotation step without a secrets manager, generate new proxy credentials in your proxy daemon's user database (e.g., htpasswd for Squid: 'htpasswd -B /etc/squid/passwd proxyuser'), update all Node.js application environment variables via your deployment config (Docker secrets, .env files, or Kubernetes Secrets), and immediately revoke the old credential at the proxy ACL level. Document the rotation timestamp for the 30-day monitoring window in Step 4.

Evidence: Before executing eradication, preserve: (1) a forensic copy of the vulnerable node_modules/axios/ directory (tar.gz with hash) from each affected application server to establish the exact vulnerable build that was running; (2) the proxy daemon's credential store (e.g., /etc/squid/passwd) with file timestamps intact to document the credential state at time of exposure; (3) any CI/CD pipeline build logs showing historical Axios version resolution across prior deployments, to establish how long the vulnerable version was present in production.

Step 4: Recovery — After patching, re-run dependency scans across all affected repositories to confirm the vulnerable Axios version is no longer present in any dependency tree, including transitive dependencies.

Validate proxy authentication flows in staging with redirect scenarios enabled. Monitor proxy and application logs for 30 days post-remediation for anomalous authentication patterns using harvested credentials. Reference: NIST AU-6 (Audit Record Review, Analysis, and Reporting); D3-LAM (Local Account Monitoring).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

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

Compensating: Run 'npm audit --audit-level=high' and 'npx better-npm-audit audit' against all repaired repositories to confirm zero Axios findings; supplement with 'npx depcheck' to catch orphaned transitive references. For 30-day post-remediation proxy monitoring without a SIEM, configure a cron job (*/15 * * * *) that greps your proxy access log for the rotated-away old proxy credential string (base64-encoded form of 'olduser:oldpassword') and emails an alert on any match — this detects replay attempts using the harvested credential. In staging, use mitmproxy's replay feature to simulate an HTTP 302 redirect mid-proxy-request and confirm the patched Axios version does not forward the Proxy-Authorization header to the redirect destination.

Evidence: Collect before declaring recovery complete: (1) 'npm ls axios --all --json' output from every production deployment confirming resolved version is patched, saved with a timestamp; (2) proxy access log snapshot from the 30-day monitoring window showing no authentication attempts using the pre-rotation credential hash; (3) staging environment test results from redirect simulation confirming header sanitization behavior of the patched Axios version under HTTP 301, 302, 307, and 308 redirect conditions.

Step 5: Post-Incident — Conduct a dependency audit to identify all third-party libraries handling HTTP headers in proxied contexts. Implement automated software composition analysis (SCA) in CI/CD pipelines to flag vulnerable library versions before deployment. Review header sanitization requirements in secure coding standards. Reference: NIST SI-4 (System Monitoring) — no mapped NIST control directly addresses SCA pipeline integration from the provided knowledge base; CIS 2.1 (Establish and Maintain a Software Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process); D3-CH (Credential Hardening).

NIST Phase: Post Incident

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

Controls: 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), NIST AU-6 (Audit Record Review, Analysis, And Reporting)

Compensating: Integrate 'npm audit' as a blocking CI step using GitHub Actions or GitLab CI with 'npm audit --audit-level=high && exit $?' — this is free and catches GHSA advisories including CVE-2026-44486 class issues at PR time. For SCA without commercial tooling, add OWASP Dependency-Check (free, CLI-based) to the pipeline: 'dependency-check.sh --project myapp --scan . --failOnCVSS 7' will block builds when a HIGH or CRITICAL npm dependency CVE is detected. Extend your secure coding standards to explicitly require that any HTTP client library used in a proxied context must be reviewed for header forwarding behavior on redirect, and add a code review checklist item: 'Does this HTTP client strip Authorization/Proxy-Authorization headers before following cross-origin redirects?'

Evidence: Document for lessons-learned record: (1) the full dependency graph output ('npm ls --all --json') from each affected application at time of discovery, archived to establish blast radius; (2) a timeline of when the vulnerable Axios version entered each repository (git log on package-lock.json filtered to axios version changes) to calculate dwell time; (3) proxy authentication logs covering the full dwell period, retained per your AU-11 retention policy, as evidence of whether credential leakage to redirect destinations occurred during that window.

Detection Guidance

Query proxy access logs and application HTTP logs for redirect chains (HTTP 301/302/307/308) where the downstream request to the redirect destination contains an Authorization or Proxy-Authorization header. Filter

specifically for cases where the redirect destination hostname differs from the configured proxy host. In Node.js application logs, look for Axios request sequences where a redirect occurs and the target URL resolves outside the expected proxy network segment. If your environment uses a centralized proxy, correlate proxy egress logs against redirect destination IPs to identify credentials forwarded to unexpected external hosts. No public IOCs are available for this vulnerability. Detection depends on header-level log visibility, which requires explicit logging configuration in most proxy and application stacks. Reference: NIST AU-3 (Content of Audit Records), ensure logs capture originating host, destination host, and header metadata for outbound HTTP requests. CIS 8.2 (Collect Audit Logs), confirm audit logging is enabled across Node.js application tiers.

Framework Mappings

MITRE-ATTACK

- **T1071.001** — Web Protocols
- **T1552** — Unsecured Credentials

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control
- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures

NIST-800-53R5

- **AC-3** — Access Enforcement
- **SC-28** — Protection of Information at Rest
- **IA-5** — Authenticator Management
- **SR-2** — Supply Chain Risk Management Plan

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control
- **164.308(a)(5)(ii)(D)** — Password Management
- **164.312(d)** — Person or Entity Authentication

CIS-V8

- **5.2** — Use Unique Passwords
- **6.3** — Require MFA for Externally-Exposed Applications
- **15.1** — Establish and Maintain an Inventory of Service Providers

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures
- **CC9.2** — Manages risks associated with vendors and business partners

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1071.001	Web Protocols	Command-And-Control
T1552	Unsecured Credentials	Credential-Access

Sources

Source	URL	Tier
osv	https://osv.dev/vulnerability/GHSA-j5f8-grm9-p9fc	T3
CVE-2026-23886 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-23886	T1
Vulnerability Directory | CVE-2026-34486 | Apache Tomcat	https://www.herodevs.com/vulnerability-directory/cve-2026-34486	T3
CVE-2026-44456: Hono Web Framework DOS Vulnerability	https://www.sentinelone.com/vulnerability-database/cve-2026-44456/	T3
CVE-2026-34486 - Red Hat Customer Portal	https://access.redhat.com/security/cve/cve-2026-34486	T3
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-44486	T1

Generated 2026-06-05 06:57 UTC by TJS Security Command Center