INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-04 19:24 UTC

GHSA-p92q-9vqr-4j8v: Axios: Proxy-Authorization Credential Leak to Origin Server Across HTTP-to-HTTPS

CVE VULNERABILITY | HIGH | CVSS 7.4

SCC Item ID	SCC-CVE-2026-0264
Type	CVE Vulnerability
CVE ID	CVE-2026-44487
Severity	HIGH
CVSS Base Score	7.4
Affected Products	axios (npm), specific versions unconfirmed from available source data
Published	2026-06-04T14:19:53Z
Discovery Source	Osv

Executive Summary

A credential leak vulnerability in the Axios HTTP client library (GHSA-p92q-9vqr-4j8v) allows proxy authentication credentials to be forwarded to unintended origin servers when an HTTP request is redirected to HTTPS. Any Node.js application using Axios with proxy authentication is potentially exposed, meaning proxy credentials, and any access they control, may be compromised. The business risk is unauthorized access to internal systems or third-party services protected by those proxy credentials.

Technical Analysis

CVE-2026-44487 (GHSA-p92q-9vqr-4j8v) affects the Axios Node.js HTTP adapter. When a request is configured with a Proxy-Authorization header and encounters an HTTP-to-HTTPS redirect, the adapter fails to strip the Proxy-Authorization header at the redirect boundary. The header is forwarded to the HTTPS origin server, exposing proxy credentials to a potentially untrusted third party. Affected package: axios (npm); specific vulnerable version range is unconfirmed from available source data, consult the OSV advisory at https://osv.dev/vulnerability/GHSA-p92q-9vqr-4j8v for version details as they are published. CWE-200 (Exposure of Sensitive Information to an Unauthorized Actor) and CWE-522 (Insufficiently Protected Credentials) apply. MITRE ATT&CK techniques T1557 (Adversary-in-the-Middle) and T1552 (Unsecured Credentials) are relevant attack patterns. EPSS score is 0.0 and the vulnerability is not currently listed in CISA KEV; exploitation in the wild has not been confirmed. Note: CVE-2026-44487 does not yet appear in NVD records; the advisory should be tracked via the OSV/GHSA identifier until NVD indexing is confirmed.

Action Checklist

1. Step 1: Containment: Identify all Node.js applications and services in your environment that import the axios npm package and are configured to use HTTP proxies with Proxy-Authorization headers. Temporarily disable proxy authentication for non-critical Axios-dependent services or route those services through a proxy that does not rely on header-based credentials, until a patched version is confirmed and deployed.
2. Step 2: Detection: Search your npm dependency manifests (package.json, package-lock.json, yarn.lock) and SBOM records for axios entries. In application logs, look for outbound HTTP-to-HTTPS redirect chains (HTTP 301/302 responses followed by HTTPS requests) originating from Node.js processes. Audit proxy server access logs for authentication attempts from unexpected origin IPs, which may indicate credential forwarding. Reference NIST AU-6 (Audit Record Review, Analysis, and Reporting) for log review process. CIS 8.2 (Collect Audit Logs) applies, confirm logging is enabled for outbound traffic from Node.js application hosts.
3. Step 3: Eradication: Monitor the OSV advisory (https://osv.dev/vulnerability/GHSA-p92q-9vqr-4j8v) and the axios GitHub repository for a patched release. Once a patched version is published, upgrade axios across all affected applications using your standard dependency update workflow. If no patch is yet available, mitigate by ensuring Axios requests to proxied destinations do not follow redirects automatically (set maxRedirects: 0 for proxy-authenticated requests) or by stripping the Proxy-Authorization header in application code before following any redirect. Reference CIS 7.4 (Perform Automated Application Patch Management) for patch deployment process.
4. Step 4: Recovery: After deploying the patched version, rotate all proxy credentials that were in use by affected applications, per D3-CRO (Credential Rotation). Verify through proxy server logs that Proxy-Authorization headers are no longer appearing in direct requests to HTTPS origin servers. Re-enable any services that were temporarily restricted in Step 1. Monitor application behavior for 72 hours post-remediation. Reference NIST AC-6 (Least Privilege), review whether affected services require broad proxy access or whether that scope can be reduced.
5. Step 5: Post-Incident: Conduct a software inventory review to identify other npm dependencies that handle proxy authentication or HTTP redirect logic, per CIS 2.1 (Establish and Maintain a Software Inventory). Evaluate whether your CI/CD pipeline includes SBOM generation and automated dependency vulnerability scanning. Implement D3-UAP (User Account Permissions) principles for proxy credentials, scope proxy credentials to the minimum set of services that require them, so that a future credential leak has limited blast radius. Document lessons learned against NIST IR controls and update your dependency management policy.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate to CISO and legal/privacy counsel immediately if proxy access logs show the leaked Proxy-Authorization credentials were used to authenticate from any IP address other than known Node.js application servers, or if those proxy credentials control access to systems processing PII, PHI, or payment data — both conditions may trigger breach notification obligations under applicable regulations.

Recovery Notes	After deploying the axios patch and rotating proxy credentials, verify remediation by running a controlled HTTP-to-HTTPS redirect test through mitmproxy and confirming the Proxy-Authorization header does not appear in the HTTPS leg of the request. Monitor upstream proxy authentication logs continuously for 72 hours post-rotation, alerting on any authentication attempt using the old credential strings, which would indicate an unpatched Node.js instance or an attacker actively replaying captured credentials. Retain pre-rotation proxy access logs for a minimum of 90 days to support any subsequent forensic review or regulatory inquiry regarding the exposure window.
Forensic Artifacts	Squid or upstream proxy access logs filtered for HTTP 301/302 responses where the subsequent request carries a Proxy-Authorization header targeting an HTTPS origin — this specific log pattern is the primary forensic signature of CVE-2026-44487 credential forwarding occurring in production traffic. | Node.js process environment snapshots (`/proc//environ` on Linux) capturing the proxy credential values that were loaded at runtime, establishing what credentials were exposed and need rotation. | axios package source file `node_modules/axios/lib/adapters/http.js` preserved from each affected application prior to patching — this file contains the redirect-handling logic where the credential leak originates and is the authoritative artifact for confirming the vulnerable code path was present. | Network packet capture (pcap) of proxy egress traffic during the exposure window, filtered for TCP streams where a `Proxy-Authorization` header in an HTTP request is followed by a TLS ClientHello to an HTTPS destination IP — this confirms whether exploitation was theoretical or active. | CI/CD pipeline dependency lock files (package-lock.json or yarn.lock) with timestamps, documenting when the vulnerable axios version was introduced into each application's dependency tree and establishing the start of the exposure window for breach notification timeline purposes.

Per-Action IR Details

Step 1: Containment — Identify all Node.js applications and services in your environment that import the axios npm package and are configured to use HTTP proxies with Proxy-Authorization headers. Temporarily disable proxy authentication for non-critical Axios-dependent services or route those services through a proxy that does not rely on header-based credentials, until a patched version is confirmed and deployed.

NIST Phase: Containment

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

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

Compensating: Run `grep -r 'axios' /path/to/projects --include='package.json' -l` across all Node.js project roots to enumerate affected applications. For each hit, inspect the instantiation for proxy config: `grep -r 'proxy\|Proxy-Authorization\|proxyAuth' /path/to/project/src`. To suppress credential forwarding immediately without patching, add `maxRedirects: 0` to all axios instances using proxy auth via a one-liner sed patch or a monkey-patch shim loaded via `NODE_OPTIONS=--require ./disable-axios-redirects.js`. On the proxy server (e.g., Squid), temporarily enforce IP allowlisting so only known app server IPs can authenticate, reducing blast radius if credentials have already leaked.

Evidence: Before making any changes, capture: (1) the output of `npm list axios --all` and `cat package-lock.json | grep -A2 '"axios"'` for each application to document the exact installed version; (2) current proxy configuration from environment variables (`printenv | grep -i proxy`) and application config files containing `proxy:` keys; (3) a snapshot of outbound network connections from Node.js processes via `ss -tnp | grep node` or `netstat -tnp | grep node` to identify active proxy-authenticated sessions in flight at time of containment.

Step 2: Detection — Search your npm dependency manifests (package.json, package-lock.json, yarn.lock) and SBOM records for axios entries. In application logs, look for outbound HTTP-to-HTTPS redirect chains

(HTTP 301/302 responses followed by HTTPS requests) originating from Node.js processes. Audit proxy server access logs for authentication attempts from unexpected origin IPs, which may indicate credential forwarding. Reference NIST AU-6 (Audit Record Review, Analysis, and Reporting) for log review process. CIS 8.2 (Collect Audit Logs) applies — confirm logging is enabled for outbound traffic from Node.js application hosts.

NIST Phase: Detection Analysis

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

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-2 (Event Logging), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Enumerate all axios installations across the environment: `find / -path '*/node_modules/axios/package.json' 2>/dev/null -exec grep -H '"version"' {} \;`. On the proxy server (Squid/Nginx), extract redirect-chain evidence: `grep -E '(301|302)' /var/log/squid/access.log | grep 'http://' | awk '{print $1,$3,$4,$7}'` — look for entries where the initial request is HTTP and the subsequent auth attempt targets an HTTPS origin. Use Wireshark or tcpdump to capture proxy egress traffic: `tcpdump -i eth0 -w /tmp/axios-proxy-capture.pcap 'tcp port 3128 or tcp port 8080'`, then filter in Wireshark for `http.request.method == "CONNECT"` followed by `Proxy-Authorization` headers appearing in requests destined for HTTPS origins. For SBOM scanning without enterprise tooling, run `npx @cyclonedx/cyclonedx-npm --output-format JSON > sbom.json` and grep for axios version strings.

Evidence: Capture before analysis concludes: (1) Squid or forward-proxy access logs filtered for HTTP 301/302 responses where the `Proxy-Authorization` header appears in a subsequent request to an HTTPS destination — the specific artifact of this CVE-2026-44487 exploit path; (2) Node.js application stdout/stderr logs showing axios request lifecycle, particularly any lines referencing redirect following (axios logs the redirect URL at debug level — enable with `axios.interceptors.request.use()` logging if not already active); (3) DNS query logs from the application host showing resolution of the HTTPS origin domain immediately after an HTTP redirect, which would confirm the credential-forwarding redirect chain was traversed.

Step 3: Eradication — Monitor the OSV advisory (https://osv.dev/vulnerability/GHSA-p92q-9vqr-4j8v) and the axios GitHub repository for a patched release. Once a patched version is published, upgrade axios across all affected applications using your standard dependency update workflow. If no patch is yet available, mitigate by ensuring Axios requests to proxied destinations do not follow redirects automatically (set maxRedirects: 0 for proxy-authenticated requests) or by stripping the Proxy-Authorization header in application code before following any redirect. Reference CIS 7.4 (Perform Automated Application Patch Management) for patch deployment process.

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), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: If a patch is not yet available, apply the redirect-suppression mitigation at scale: use a project-wide sed command to inject `maxRedirects: 0` into all axios instantiations — `grep -rl 'axios.create\|axios({' ./src | xargs sed -i 's/axios.create({/axios.create({ maxRedirects: 0,/g'` — then verify with `grep -r 'maxRedirects' ./src`. Alternatively, deploy a global axios request interceptor via a shared module: `axios.interceptors.request.use(config => { if (config.proxy) { config.maxRedirects = 0; } return config; });`. Once a patched version releases, upgrade using `npm install axios@ --save-exact` and lock the version in package.json to prevent regression. Validate the patch removed the credential-forwarding behavior by testing with mitmproxy: send an HTTP request through an authenticated proxy to a destination that issues a 301 redirect to HTTPS and confirm `Proxy-Authorization` does not appear in the HTTPS leg.

Evidence: Before eradication, preserve: (1) a full copy of the vulnerable axios package directory (`cp -r ./node_modules/axios /forensics/axios-pre-patch/`) including `lib/adapters/http.js`, which contains the redirect-handling code responsible for this credential leak — this is the file that would be modified by the patch and must be preserved for comparison; (2) a record of the exact axios version string from `node_modules/axios/package.json` for each application; (3) any CI/CD pipeline dependency lock files (package-lock.json, yarn.lock) prior to update, preserved as forensic baseline artifacts documenting the vulnerable dependency state.

Step 4: Recovery — After deploying the patched version, rotate all proxy credentials that were in use by affected applications, per D3-CRO (Credential Rotation). Verify through proxy server logs that Proxy-Authorization headers are no longer appearing in direct requests to HTTPS origin servers. Re-enable any services that were temporarily restricted in Step 1. Monitor application behavior for 72 hours post-remediation. Reference NIST AC-6 (Least Privilege) — review whether affected services require broad proxy access or whether that scope can be reduced.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AC-6 (Least Privilege), NIST AC-2 (Account Management), CIS 5.2 (Use Unique Passwords), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Rotate proxy credentials immediately after patch deployment: update the proxy password in the upstream proxy server (e.g., in Squid's `passwd` file via `htpasswd -b /etc/squid/passwd ` and `squid -k reconfigure`), then update the new credentials in each application's environment variables or secrets store and restart affected Node.js processes. To verify the patch is effective, run a controlled test using mitmproxy or Burp Suite in transparent mode: issue an axios request with proxy auth to an HTTP endpoint that 301-redirects to HTTPS and confirm via captured traffic that the `Proxy-Authorization` header is absent from the HTTPS request. Monitor proxy access logs for 72 hours using: `tail -f /var/log/squid/access.log | grep -i 'proxy-authorization'` — any appearance of the old credential string after rotation indicates an unpatched instance remains active.

Evidence: Before re-enabling restricted services, capture: (1) proxy server authentication logs showing all credential usage in the 30 days prior to credential rotation, to establish whether the old credentials were used by unauthorized parties or from unexpected source IPs — this is the primary indicator that CVE-2026-44487 was actively exploited rather than merely present; (2) a post-patch network capture (`tcpdump -i eth0 -w /tmp/axios-post-patch.pcap 'tcp port 443'`) during a controlled redirect-chain test, confirming `Proxy-Authorization` is absent from HTTPS legs; (3) application restart timestamps and process IDs confirming all Node.js instances were restarted with the new axios version and rotated credentials.

Step 5: Post-Incident — Conduct a software inventory review to identify other npm dependencies that handle proxy authentication or HTTP redirect logic, per CIS 2.1 (Establish and Maintain a Software Inventory). Evaluate whether your CI/CD pipeline includes SBOM generation and automated dependency vulnerability scanning. Implement D3-UAP (User Account Permissions) principles for proxy credentials — scope proxy credentials to the minimum set of services that require them, so that a future credential leak has limited blast radius. Document lessons learned against NIST IR controls and update your dependency management policy.

NIST Phase: Post Incident

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

Controls: CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.3 (Address Unauthorized Software), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), NIST AU-11 (Audit Record Retention), NIST AC-6 (Least Privilege)

Compensating: Extend SBOM coverage to catch similar proxy-handling vulnerabilities in other npm packages: run `npm audit` across all Node.js projects and additionally query OSV directly for any package that wraps Node's `http.request` or handles redirects — `npx better-npm-audit audit --level high`. Integrate automated SBOM generation into CI/CD using `npx @cyclonedx/cyclonedx-npm` as a pipeline step with a GitHub Actions or Jenkins post-build action that fails the build on HIGH/CRITICAL advisories matching proxy or redirect keyword patterns. For credential scoping, implement per-service proxy accounts on the upstream proxy (distinct Squid ACL entries per application source IP) so that a future axios-class credential leak from one service cannot be used to authenticate as another service. Document the lessons-learned finding that HTTP-to-HTTPS redirect chains in proxied Node.js applications represent a credential exfiltration vector requiring explicit `maxRedirects` governance in your secure coding standard.

Evidence: Retain for post-incident review: (1) the full proxy access log archive covering the window from when the vulnerable axios version was first deployed to when credentials were rotated — this defines the maximum exposure window for CVE-2026-44487 and is required for any breach notification assessment; (2) the pre-patch SBOM snapshots captured during eradication, which document the exact dependency state of each application at time of

incident and serve as the baseline for measuring remediation completeness; (3) output of `npm audit` run against all Node.js applications post-patch, saved as a timestamped artifact confirming no remaining HIGH/CRITICAL advisories related to axios or proxy-handling dependencies.

Detection Guidance

Primary detection surface is outbound HTTP traffic from Node.js application hosts. Query proxy server access logs for requests that include a Proxy-Authorization header directed at HTTPS endpoints (port 443) rather than the proxy itself, this is the leak pattern. In application-layer logs, look for HTTP 3xx redirect responses where the subsequent request crosses an HTTP-to-HTTPS boundary. In your SIEM or log aggregation platform, correlate Node.js process outbound connections: flag any session where an initial HTTP request (port 80) is followed within the same session by an HTTPS request (port 443) to a different host, with no header-stripping event logged. Search package manifests using 'npm list axios' or equivalent across your asset inventory to identify affected hosts. No public IOCs (IPs, domains, hashes) are associated with this vulnerability at this time, exploitation is opportunistic and server-side, leaving traces in proxy and origin server logs rather than on the endpoint. Reference NIST AU-2 (Event Logging) to confirm outbound HTTP/HTTPS traffic is captured in your logging scope. D3-LAM (Local Account Monitoring) applies if proxy credentials are tied to service accounts, monitor those accounts for unexpected authentication activity.

Framework Mappings

MITRE-ATTACK

- **T1557** — Adversary-in-the-Middle
- **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
- **SI-4** — System Monitoring

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1557	Adversary-in-the-Middle	Credential-Access
T1552	Unsecured Credentials	Credential-Access

Sources

Source	URL	Tier
osv	https://osv.dev/vulnerability/GHSA-p92q-9vqr-4j8v	T3
CVE-2023-44487 Detail - NVD	https://nvd.nist.gov/vuln/detail/cve-2023-44487	T1
HTTP/2 Rapid Reset Vulnerability, CVE-2023-44487 | CISA	https://www.cisa.gov/news-events/alerts/2023/10/10/http2-rapid-rese...	T1
CVE 2023-44487 and How To Avoid an HTTP/2 Rapid Reset Attack	https://www.openlogic.com/blog/cve-2023-44487-http-2-rapid-reset	T3
K000137106: HTTP/2 vulnerability CVE-2023-44487 - MyF5 | Support	https://my.f5.com/manage/s/article/K000137106	T3
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-44487	T1

DISCLAIMER

This intelligence report is produced by Tech Jacks Solutions Security Command Center (SCC) for informational purposes only. It does not constitute professional security advice, legal counsel, or an incident response engagement. The information herein is derived from publicly available sources and AI-assisted analysis; while every effort is made to ensure accuracy, Tech Jacks Solutions makes no warranties regarding completeness or timeliness. Organizations should conduct their own validation and consult qualified security professionals before taking action based on this report. Tech Jacks Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-06-04 19:24 UTC by TJS Security Command Center