INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-12 19:07 UTC

GHSA-mg66-mrh9-m8jx: Next.js vulnerable to Denial of Service via connection exhaustion in application

CVE VULNERABILITY | **MEDIUM** | CVSS 5.9

SCC Item ID	SCC-CVE-2026-0164
Type	CVE Vulnerability
CVE ID	CVE-2026-44579
Severity	MEDIUM
CVSS Base Score	5.9
Affected Products	next (npm), specific affected versions not confirmed from available sources
Published	2026-05-11T15:56:24Z
Discovery Source	Osv

Executive Summary

A denial-of-service vulnerability in Next.js (GHSA-mg66-mrh9-m8jx, CVE-2026-44579) allows attackers to exhaust connection pools in applications using Cache Components, taking those applications offline. Organizations running Next.js in customer-facing or internal web applications are potentially exposed. Affected version ranges and patch availability have not been confirmed from authoritative sources at this time, so exposure assessment requires direct verification against your deployed versions.

Technical Analysis

CVE-2026-44579 (GHSA-mg66-mrh9-m8jx) describes a Denial of Service condition in Next.js (npm package: next) rooted in CWE-770 (Allocation of Resources Without Limits or Throttling) and CWE-400 (Uncontrolled Resource Consumption). The attack vector targets the Cache Components subsystem, where an attacker can trigger connection exhaustion, overwhelming the application's connection pool or resource allocation mechanism until the service becomes unavailable. MITRE ATT&CK mapping: T1499 (Endpoint Denial of Service), T1499.002 (Service Exhaustion Flood). CVSS base score: 5.9 (Medium) from NVD provisional assessment; Vercel has not yet published an independent CVSS score. CVSS vector pending full NVD publication. EPSS score and percentile are both 0.0 at analysis time, indicating low current exploitation probability. Not listed in CISA KEV. Important data gap: specific affected version ranges and confirmed patch availability were not verifiable from accessible authoritative sources at analysis time. Primary source identifier is OSV record GHSA-mg66-mrh9-m8jx. Note: CVE-2026-4579 (without the middle zero) references a separate

SQLi vulnerability in Simple Laundry System and is NOT related to CVE-2026-44579 (this Next.js advisory). Do not confuse the two identifiers. Analysts should monitor the Next.js GitHub advisory page and npm security advisories for version-specific remediation guidance.

Action Checklist

1. Step 1: Containment, Identify all production deployments of the Next.js npm package using Cache Components. Rate-limit or restrict inbound connections to Next.js application endpoints at the load balancer or WAF layer to reduce connection exhaustion exposure while patch status is confirmed. Prioritize internet-facing instances.
2. Step 2: Detection, Query your software inventory or SCA tooling for the next npm package across all environments. Review application and infrastructure logs for abnormal connection pool saturation events, sudden spikes in open connections, or service unavailability patterns correlated with inbound request surges. There are no confirmed IOCs for this vulnerability at this time.
3. Step 3: Eradication, Specific affected version ranges and an official patch have not been confirmed from authoritative sources at analysis time. Monitor the Next.js GitHub security advisories (github.com/vercel/next.js/security/advisories) and the OSV record GHSA-mg66-mrh9-m8jx for patch release. Apply the vendor-issued patch or version upgrade as soon as it is published and validated.
4. Step 4: Recovery, After applying any vendor-issued remediation, verify that Cache Components behave normally under load. Monitor connection pool metrics and error rates for at least 24 hours post-remediation. Confirm that connection counts return to baseline under normal traffic.
5. Step 5: Post-Incident, Review whether connection pool limits and resource throttling controls are defined and enforced across all Next.js deployments. Evaluate whether your software composition analysis process would have flagged this package before deployment. Add GHSA-mg66-mrh9-m8jx to your vulnerability tracking backlog for version confirmation and patch validation when authoritative details become available.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate to incident commander and notify application owners immediately if connection pool saturation is actively observed (ESTAB socket count on Next.js host exceeds 2x baseline, or HTTP 503 error rate exceeds 5% on Cache Component routes), if the application processes PII or PHI triggering availability-based breach notification obligations, or if affected version ranges are confirmed and no patch is available within 72 hours of advisory publication.
Recovery Notes	After applying the vendor-issued patch for CVE-2026-44579, run a controlled load test against the specific Next.js Cache Component endpoints that were vulnerable to connection exhaustion — not just a generic health check — to confirm the fix closes the resource pool leak under concurrent request conditions. Monitor 'ss -s' ESTAB and CLOSE-WAIT counts plus application-layer HTTP 503 and 504 error rates at 1-hour intervals for a minimum of 24 hours post-deployment, as connection handle leaks can manifest slowly under moderate traffic. Do not remove load balancer rate-limiting controls applied during containment until at least 24 hours of clean baseline metrics have been confirmed post-patch.

Forensic Artifacts	Node.js runtime logs (stdout/stderr from the Next.js server process) — look for repeated 'ECONNRESET', 'socket hang up', or connection timeout errors originating from Cache Component fetch operations, which indicate the pool was being exhausted by inbound request volume | Load balancer / reverse proxy access logs (nginx access.log, AWS ALB access logs, or Cloudflare logs) — filter for HTTP 503/504 responses on routes served by Cache Components correlated with inbound request-per-second spikes, providing a timeline of when exhaustion attempts began | Socket state snapshots from 'ss -s' or 'netstat -an' captured at time of incident — CLOSE-WAIT accumulation on the Next.js process port is a specific artifact of connection pool exhaustion where the server side holds sockets open after client disconnects, distinguishing this from generic traffic spikes | package-lock.json and 'npm list next --depth=0' output from each affected host — documents the exact installed version of the 'next' npm package at time of incident, required for advisory version-range confirmation once authoritative details are published for CVE-2026-44579 | .next/cache/ directory metadata and Next.js server-side cache configuration files — documents whether Cache Components were actively in use and how they were configured, establishing that the vulnerable feature was enabled and scoping which routes were exposed to the connection exhaustion vector

Per-Action IR Details

Step 1: Containment — Identify all production deployments of the Next.js npm package using Cache Components. Rate-limit or restrict inbound connections to Next.js application endpoints at the load balancer or WAF layer to reduce connection exhaustion exposure while patch status is confirmed. Prioritize internet-facing instances.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: short-term containment to limit damage while preserving options for eradication

Controls: NIST IR-4 (Incident Handling), NIST SC-5 (Denial-of-Service Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: On nginx or Apache acting as reverse proxy in front of Next.js: set 'limit_conn' (nginx) or 'LimitRequestFields' (Apache) to cap concurrent connections per source IP before they reach the Next.js process. On AWS ALB without WAF budget, use a Security Group inbound rule to restrict source CIDR to known trusted ranges for internal-only applications. Run 'ss -s' or 'netstat -an | grep ESTABLISHED | wc -l' on the Next.js host every 5 minutes via cron to detect connection count growth before full exhaustion occurs.

Evidence: Before applying rate limits, capture a baseline snapshot of current connection state with 'ss -tunp | grep node' (or the Next.js server process PID) to document the normal open-connection count. Export load balancer access logs covering the 48 hours prior to detection — these will show the source IP distribution and request-per-second rate needed to determine whether exhaustion attempts have already begun against Cache Component endpoints.

Step 2: Detection — Query your software inventory or SCA tooling for the next npm package across all environments. Review application and infrastructure logs for abnormal connection pool saturation events, sudden spikes in open connections, or service unavailability patterns correlated with inbound request surges. There are no confirmed IOCs for this vulnerability at this time.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: identify precursors and indicators, determine scope, and document affected systems

Controls: NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-2 (Flaw Remediation), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 8.2 (Collect Audit Logs)

Compensating: Enumerate all environments for the 'next' package version without a commercial SCA tool: run 'find / -name package.json -not -path "*/node_modules/*/node_modules/*" 2>/dev/null | xargs grep -l "\"next\"" on each host,

then extract the resolved version with 'cat package-lock.json | python3 -c "import sys,json; d=json.load(sys.stdin); print(d.get(\"packages\",{}).get(\"node_modules/next\",{}).get(\"version\",\"not found\"))"'. For connection saturation detection without a SIEM, deploy a bash script via cron every 60 seconds that runs 'ss -s' and appends the output with a timestamp to a local log file, alerting via email if ESTAB count exceeds a defined threshold (e.g., 500).

Evidence: Collect Node.js process-level metrics from the application runtime (stdout/stderr logs showing 'ECONNRESET', 'ETIMEDOUT', or connection refused errors tied to Cache Component fetch calls). Pull Next.js server logs (default path: '.next/server/') for request queuing errors and cache miss storms. Export load balancer access logs and filter for HTTP 503 or 504 responses correlated with high-concurrency request bursts to Cache Component routes — this pattern is the fingerprint of connection pool exhaustion rather than a generic outage.

Step 3: Eradication — Specific affected version ranges and an official patch have not been confirmed from authoritative sources at analysis time. Monitor the Next.js GitHub security advisories (github.com/vercel/next.js/security/advisories) and the OSV record GHSA-mg66-mrh9-m8jx for patch release. Apply the vendor-issued patch or version upgrade as soon as it is published and validated.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication: eliminate components of the incident and identify and mitigate all exploited vulnerabilities

Controls: NIST SI-2 (Flaw Remediation), NIST SI-5 (Security Alerts, Advisories, and Directives), NIST CM-3 (Configuration Change Control), CIS 7.4 (Perform Automated Application Patch Management), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Subscribe to the Vercel Next.js GitHub security advisory feed via RSS (github.com/vercel/next.js/security/advisories.atom) and the OSV.dev feed for GHSA-mg66-mrh9-m8jx to receive patch notifications without a commercial vulnerability management platform. Once a patch is published, test it in a staging environment running the same Cache Component configuration as production before promoting; use 'npm audit' post-upgrade to confirm the advisory is resolved. If a patch is delayed, evaluate whether Cache Components can be temporarily disabled or replaced with server-side rendering for the affected routes as an interim architectural mitigation.

Evidence: Before patching, preserve the exact 'package.json', 'package-lock.json', and '.next/cache/' directory state from each affected deployment as forensic evidence of the vulnerable configuration. Document the installed 'next' package version from each host using 'npm list next --depth=0' and retain this output alongside the advisory record to support post-incident timeline reconstruction and any required regulatory disclosure.

Step 4: Recovery — After applying any vendor-issued remediation, verify that Cache Components behave normally under load. Monitor connection pool metrics and error rates for at least 24 hours post-remediation. Confirm that connection counts return to baseline under normal traffic.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery: restore systems to normal operation and verify that systems are functioning normally

Controls: NIST IR-4 (Incident Handling), NIST SI-6 (Security and Privacy Function Verification), NIST CP-10 (System Recovery and Reconstitution), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Use Apache JMeter (free) or 'hey' (open-source HTTP load generator) to simulate concurrent request load against the Next.js Cache Component endpoints post-patch and confirm that connection counts stabilize rather than climbing unboundedly. Monitor the Node.js process with 'node --inspect' or by polling 'process.memoryUsage()' and 'process._getActiveHandles().length' via a lightweight health-check endpoint to verify handle counts return to pre-incident baseline. Set up a 24-hour cron job that logs 'ss -s' output every 5 minutes to a timestamped file for post-recovery evidence retention.

Evidence: Capture post-patch connection pool metrics immediately after deployment and at 1-hour intervals for 24 hours — specifically the ESTAB, TIME-WAIT, and CLOSE-WAIT socket counts from 'ss -s' on the Next.js host — to demonstrate that the exhaustion vector is closed. Retain load balancer access logs showing the return to normal HTTP 200 response rates on Cache Component routes as documented evidence of successful remediation.

Step 5: Post-Incident — Review whether connection pool limits and resource throttling controls are defined and enforced across all Next.js deployments. Evaluate whether your software composition analysis process would have flagged this package before deployment. Add GHSA-mg66-mrh9-m8jx to your vulnerability tracking backlog for version confirmation and patch validation when authoritative details become available.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: lessons learned, update policies, improve detection, and share intelligence

Controls: NIST IR-4 (Incident Handling), NIST IR-5 (Incident Monitoring), NIST IR-8 (Incident Response Plan), NIST SI-2 (Flaw Remediation), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: If no commercial SCA tooling is in place, integrate 'npm audit' into your CI/CD pipeline (GitHub Actions, GitLab CI, or Jenkins) as a mandatory pre-deploy gate that fails builds on high/critical advisories and logs medium advisories — this would surface GHSA-mg66-mrh9-m8jx automatically on next build. Add an OSV-Scanner GitHub Action (github.com/google/osv-scanner) to scan 'package-lock.json' on every pull request, specifically covering the npm ecosystem to catch Next.js and related Vercel package advisories before they reach production.

Evidence: Retain the full incident timeline document including: the date GHSA-mg66-mrh9-m8jx was published versus the date it was detected in your environment (gap analysis), the list of affected Next.js deployments identified in Step 2, the connection pool baseline and saturation metrics captured during the incident, and the patch application date with 'npm list next' output confirming remediated versions — this record supports both internal lessons-learned and any regulatory notification obligations if availability impact affected regulated workloads.

Detection Guidance

No confirmed IOCs or exploitation indicators are available for this vulnerability at analysis time. Detection should focus on anomaly-based signals: monitor application connection pool metrics for sustained exhaustion events, watch for HTTP 503 or connection timeout error spikes correlated with inbound traffic surges to Next.js endpoints, and review infrastructure-level connection count metrics at the load balancer or reverse proxy. In your SCA or dependency management tooling, query for the next npm package and flag any deployments using Cache Components. If you have WAF logging, review for repeated requests targeting Cache Component endpoints. Set up alerting on the OSV record GHSA-mg66-mrh9-m8jx and Next.js GitHub security advisories for version and patch details as they become available.

Framework Mappings

MITRE-ATTACK

- **T1499** — Endpoint Denial of Service
- **T1499.002** — Service Exhaustion Flood

NIST-800-53R5

- **SC-5** — Denial-of-Service Protection

CIS-V8

- **13.8** — Deploy a Network Intrusion Prevention Solution
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1499	Endpoint Denial of Service	Impact
T1499.002	Service Exhaustion Flood	Impact

Sources

Source	URL	Tier
osv	https://osv.dev/vulnerability/GHSA-mg66-mrh9-m8jx	T3
(consolidated)	https://osv.dev/vulnerability/GHSA-8h8q-6873-q5fj	T3
CVE-2026-4579 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-4579	T1
CVE-2026-4579: Simple Laundry System SQLi Vulnerability	https://www.sentinelone.com/vulnerability-database/cve-2026-4579/	T3
Known Exploited Vulnerabilities Catalog | CISA	https://www.cisa.gov/known-exploited-vulnerabilities-catalog	T1
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-44579	T1

Generated 2026-05-12 19:07 UTC by TJS Security Command Center