INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-12 06:37 UTC

GHSA-267c-6grr-h53f: Next.js has a Middleware / Proxy bypass in App Router applications via segment-p

CVE VULNERABILITY | HIGH | CVSS 8.2

SCC Item ID	SCC-CVE-2026-0156
Type	CVE Vulnerability
CVE ID	CVE-2026-44575
Severity	HIGH
CVSS Base Score	8.2
Affected Products	next (npm), specific affected versions not confirmed from available sources; review GHSA-267c-6grr-h53f for version ranges
Published	2026-05-11T15:54:24Z
Discovery Source	Osv

Executive Summary

A high-severity authentication bypass vulnerability (CVE-2026-44575, CVSS 8.2) affects Next.js applications using the App Router, allowing attackers to circumvent middleware-enforced authentication and authorization controls. Any organization running Next.js-based web applications, including customer portals, APIs, and internal tools built on this framework, may be exposing protected resources to unauthenticated access. Cloudflare deployed an emergency WAF rule on 2026-05-07 for this vulnerability, suggesting active exploitation risk; organizations using Cloudflare should activate this rule immediately.

Technical Analysis

CVE-2026-44575 (GHSA-267c-6grr-h53f) is a middleware and proxy bypass vulnerability in Next.js App Router applications. The flaw is rooted in how segment-prefetch routes are processed, enabling an attacker to craft requests that bypass middleware logic, including authentication gates and authorization checks, implemented in Next.js middleware. Weaknesses are classified under CWE-288 (Authentication Bypass Using an Alternate Path) and CWE-863 (Incorrect Authorization), as identified in GHSA-267c-6grr-h53f on OSV.dev. MITRE techniques T1556 (Modify Authentication Process) and T1190 (Exploit Public-Facing Application) apply. Affected version ranges are available in GHSA-267c-6grr-h53f and NVD CVE-2026-44575 records; consult those sources before patching. Related variants are tracked in consolidated advisories GHSA-26hh-7cqf-hhc6 and GHSA-36qx-fr4f-26g5. CISA KEV listing is not confirmed as of this report. EPSS data is not yet populated.

Cloudflare deployed an emergency WAF rule on 2026-05-07, indicating the attack surface is well-defined and exploitation is feasible without significant barriers.

Action Checklist

1. Step 1: Containment, Identify all Next.js App Router applications in your environment. If running Cloudflare, verify the emergency WAF rule (released 2026-05-07) is active and applied to relevant zones. For applications without WAF coverage, consider temporarily blocking or requiring network-layer authentication for routes protected by Next.js middleware until a patch is applied. Reference: Cloudflare Community WAF Release 2026-05-07.
2. Step 2: Detection, Review web server and application access logs for requests targeting segment-prefetch routes (look for prefetch-related path segments or headers, e.g., Next-Router-Prefetch or _rsc parameters) that reach protected routes without corresponding authenticated session tokens. Query your SIEM for HTTP 200 responses to routes that should return 401/403 for unauthenticated users. Check CDN and WAF logs for rule matches on the Cloudflare emergency WAF rule deployed 2026-05-07.
3. Step 3: Eradication, Apply the patched Next.js version as soon as it is published to the npm registry under the 'next' package. Check the Next.js security advisory or GHSA-267c-6grr-h53f on OSV.dev for the confirmed fixed version. If a patch is not yet available, prioritize WAF deployment or network-layer access controls as compensating controls. Do not rely solely on WAF rules as a permanent fix; they are compensating controls, not remediation.
4. Step 4: Recovery, After upgrading, verify that middleware authentication and authorization logic is enforced correctly by running authenticated and unauthenticated test requests against previously protected routes. Confirm no 200-class responses are returned for unauthenticated access to protected App Router paths. Continue monitoring WAF and application logs for anomalous prefetch-route access patterns for at least 72 hours post-patch.
5. Step 5: Post-Incident, Audit middleware implementation patterns across all Next.js applications to ensure authorization controls are not solely middleware-dependent without defense-in-depth at the API or data layer. Document compensating control gaps where WAF coverage is absent. Consider adding automated tests that verify protected routes reject unauthenticated requests after every Next.js dependency update.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO and legal/compliance if Step 2 log analysis confirms HTTP 200 responses to unauthenticated prefetch-bypass requests against routes handling PII, PHI, financial data, or authenticated user content — indicating active exploitation that may trigger breach notification obligations under GDPR, HIPAA, or applicable state privacy law.

Recovery Notes	After applying the patched Next.js version confirmed via GHSA-267c-6grr-h53f on OSV.dev, execute the `curl`-based prefetch-header bypass tests against every protected App Router route to verify middleware enforcement is restored at the application layer — do not treat Cloudflare WAF rule match absence as sufficient proof. Maintain heightened monitoring of web server and WAF logs for `_rsc`, `RSC: 1`, and `Next-Router-Prefetch` header patterns for a minimum of 72 hours post-patch, as threat actors aware of the vulnerability may continue probing for unpatched instances or attempt to exploit the window between WAF rule deployment and patch application. If log analysis from Step 2 identified confirmed exploitation events, extend the monitoring window to 30 days and conduct a full review of data accessed via the bypassed routes.
Forensic Artifacts	Web server access logs (nginx /var/log/nginx/access.log or Apache /var/log/apache2/access.log): Filter for requests containing `_rsc=` query parameters or `Next-Router-Prefetch: 1` and `RSC: 1` HTTP headers targeting authenticated App Router paths (e.g., /dashboard, /admin, /api/*) that returned HTTP 200 — these are the fingerprint of successful CVE-2026-44575 bypass exploitation. | Cloudflare Firewall Events log: Export all events matching the emergency WAF rule deployed 2026-05-07 for the 30-day window preceding deployment to identify exploitation attempts against the organization before the compensating control was in place, including source IPs and targeted URIs. | Next.js runtime / application server logs (PM2 logs via `pm2 logs`, container stdout, or systemd journal via `journalctl -u nextjs`): Capture any middleware skip, route resolution anomalies, or unhandled request events that may indicate the segment-prefetch bypass was exercised at the framework layer. | npm package artifact evidence: Preserved `package.json`, `package-lock.json`, and file-level SHA-256 hashes of `node_modules/next/dist/server/` middleware files at time of incident, documenting the vulnerable version state for chain-of-custody and post-incident timeline reconstruction. | Application session and authentication logs: Records correlating unauthenticated requests (absence of valid session cookie or Authorization header) with HTTP 200 responses on protected routes during the exploitation window — these establish whether data exposure or unauthorized access to protected resources occurred, directly informing breach notification assessment.

Per-Action IR Details

Step 1: Containment — Identify all Next.js App Router applications in your environment. If running Cloudflare, verify the emergency WAF rule (released 2026-05-07) is active and applied to relevant zones. For applications without WAF coverage, consider temporarily blocking or requiring network-layer authentication for routes protected by Next.js middleware until a patch is applied. Reference: Cloudflare Community WAF Release 2026-05-07.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST SI-4 (System Monitoring), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Run `find / -name 'package.json' -path '*/node_modules/next/package.json' 2>/dev/null | xargs grep -l '"version"'` across all servers to enumerate Next.js deployments and version strings. For applications without Cloudflare, add an nginx `location` block or Apache `RewriteRule` to block requests containing prefetch-related headers (`Next-Router-Prefetch`, `RSC: 1`, `_rsc=` query parameters) to authenticated routes, returning HTTP 403. Document each application and its WAF coverage status in a spreadsheet before proceeding.

Evidence: Before blocking, capture a full snapshot of current web server access logs (nginx: `/var/log/nginx/access.log`; Apache: `/var/log/apache2/access.log`) and any CDN/WAF edge logs for the 30 days preceding containment action. Preserve raw log files with `sha256sum` checksums for chain-of-custody. If Cloudflare is

in use, export the Firewall Events log filtered to the period before 2026-05-07 WAF rule deployment to establish a pre-rule baseline of prefetch-route traffic reaching origin.

Step 2: Detection — Review web server and application access logs for requests targeting segment-prefetch routes (look for prefetch-related path segments or headers, e.g., Next-Router-Prefetch or _rsc parameters) that reach protected routes without corresponding authenticated session tokens. Query your SIEM for HTTP 200 responses to routes that should return 401/403 for unauthenticated users. Check CDN and WAF logs for rule matches on the Cloudflare emergency WAF rule deployed 2026-05-07.

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 SI-4 (System Monitoring), NIST IR-5 (Incident Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: Without a SIEM, run the following against nginx access logs to surface HTTP 200 responses on protected routes that included `_rsc` or `Next-Router-Prefetch` indicators: `grep -E '(_rsc=|Next-Router-Prefetch)' /var/log/nginx/access.log | awk '$9 == 200' | sort | uniq -c | sort -rn`. For Apache: substitute `/var/log/apache2/access.log`. Use `grep -v` to filter out known authenticated session IP ranges. If application logs capture session tokens or auth headers, correlate HTTP 200 responses on routes like `/dashboard`, `/admin`, `/api/user` against requests lacking a valid `Set-Cookie` or `Authorization` header. A Sigma rule targeting `cs-uri-stem` containing `_rsc` with `sc-status: 200` and absence of auth cookie can be run against exported IIS/W3C logs via sigma-cli with the EVTX or plaintext log backend.

Evidence: Capture: (1) Web server access logs filtered for requests with `Next-Router-Prefetch: 1` or `RSC: 1` HTTP headers and `_rsc` query parameters, noting source IPs, timestamps, and HTTP response codes. (2) Application-layer session logs showing which requests received HTTP 200 on routes that enforce middleware authentication, cross-referenced against session token presence. (3) Cloudflare Firewall Events export showing rule match timestamps for the emergency WAF rule, to identify exploitation attempts that predated the 2026-05-07 rule deployment. (4) Any server-side error logs (Next.js runtime logs, PM2 logs, or container stdout) for unexpected route resolution or middleware skip events around the exploitation window.

Step 3: Eradication — Apply the patched Next.js version as soon as it is published to the npm registry under the 'next' package. Consult GHSA-267c-6grr-h53f on OSV.dev for the confirmed fixed version range before upgrading. Do not rely solely on WAF rules as a permanent fix — they are a compensating control, not a remediation.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CM-3 (Configuration Change Control), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: Before upgrading, run `npm audit --json | jq '.vulnerabilities.next'` to confirm the installed version is in the vulnerable range per GHSA-267c-6grr-h53f. After identifying the fixed version on OSV.dev (https://osv.dev/vulnerability/GHSA-267c-6grr-h53f — verify this URL resolves before use), execute `npm install next@ --save-exact` and lock with `package-lock.json`. Verify the installed version post-upgrade with `cat node_modules/next/package.json | jq '.version'`. Run `npm audit` post-upgrade to confirm GHSA-267c-6grr-h53f no longer appears. For teams using Docker, rebuild the container image from scratch with the patched `next` version and redeploy rather than patching in-place.

Evidence: Before patching, preserve: (1) A copy of the current `package.json` and `package-lock.json` (or `yarn.lock`) as forensic evidence of the vulnerable version in production. (2) A directory listing with file hashes (`sha256sum`) of `node_modules/next/dist/server/` — specifically middleware-related files such as `next-server.js` and any files in `dist/server/app/` — to document the vulnerable artifact state. (3) Runtime process list (`ps aux` or `pm2 list`) showing the Next.js process and its start time, to bound the window during which the vulnerable version was running.

Step 4: Recovery — After upgrading, verify that middleware authentication and authorization logic is enforced correctly by running authenticated and unauthenticated test requests against previously protected routes. Confirm no 200-class responses are returned for unauthenticated access to protected App Router paths. Continue monitoring WAF and application logs for anomalous prefetch-route access patterns for at least 72 hours post-patch.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-6 (Security and Privacy Function Verification), NIST CA-7 (Continuous Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Use `curl` to run unauthenticated bypass verification tests against each protected App Router route, explicitly sending the prefetch headers that trigger the CVE-2026-44575 bypass: `curl -v -H 'Next-Router-Prefetch: 1' -H 'RSC: 1' https:///protected-route` — a patched application must return HTTP 401 or 403, not 200. Script this across all protected routes with `while read route; do echo "Testing $route:"; curl -s -o /dev/null -w "%{http_code}" -H 'Next-Router-Prefetch: 1' https://$route; echo; done < protected_routes.txt`. For 72-hour monitoring without SIEM, configure a cron job every 15 minutes to run `grep -c '_rsc=' /var/log/nginx/access.log` and alert if counts exceed baseline, using a simple threshold script that emails or Slacks the on-call.

Evidence: Capture test results from both authenticated and unauthenticated `curl` requests against protected routes as timestamped evidence that the patched version correctly enforces middleware. Retain post-upgrade `npm audit` output confirming GHSA-267c-6grr-h53f resolution. Archive the 72-hour post-patch WAF and web server logs as evidence of no recurrence or renewed exploitation attempts during the recovery observation window.

Step 5: Post-Incident — Audit middleware implementation patterns across all Next.js applications to ensure authorization controls are not solely middleware-dependent without defense-in-depth at the API or data layer. Document compensating control gaps where WAF coverage is absent. Consider adding automated tests that verify protected routes reject unauthenticated requests after every Next.js dependency update.

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 SI-2 (Flaw Remediation), NIST RA-5 (Vulnerability Monitoring and Scanning), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 6.3 (Require MFA for Externally-Exposed Applications)

Compensating: Conduct a code review of all Next.js App Router `middleware.ts` files to identify routes where authorization is enforced only at the middleware layer with no redundant server-side check in the API route handler or React Server Component. For each such pattern, add a server-side auth check (e.g., `getServerSession()` or equivalent) as defense-in-depth. Integrate `npm audit` into CI/CD pipelines (GitHub Actions: `- run: npm audit --audit-level=high`) so future Next.js updates triggering HIGH or CRITICAL advisories block deployment automatically. Use `osquery` (`SELECT name, version FROM npm_packages WHERE name = 'next'`) for ongoing asset-level version tracking across the fleet.

Evidence: Produce a post-incident report documenting: (1) The full list of Next.js applications in the environment and their WAF coverage status at the time of disclosure. (2) Evidence of any HTTP 200 responses to prefetch-bypass requests captured in Step 2 logs, which would confirm exploitation occurred versus exposure only. (3) The timeline from CVE-2026-44575 disclosure to Cloudflare WAF rule deployment (2026-05-07) to patch application, to calculate organizational response window for future SLA benchmarking. (4) Code audit findings showing which applications had middleware-only authorization patterns, for remediation tracking.

Detection Guidance

Query access logs and SIEM for HTTP requests to App Router paths that include prefetch indicators, specifically headers such as Next-Router-Prefetch: 1 or RSC-related query parameters (_rsc), that result in

HTTP 200 responses on routes configured to require authentication. A bypass would manifest as a successful response to a protected route without a valid session cookie or authorization token. Cross-reference against your application's defined protected route patterns. If Cloudflare WAF is in use, search WAF logs for matches on the emergency rule released 2026-05-07 to identify exploitation attempts. Absence of WAF rule matches does not confirm safety; non-Cloudflare environments lack this signal.

Framework Mappings

MITRE-ATTACK

- **T1556** — Modify Authentication Process
- **T1190** — Exploit Public-Facing Application

NIST-800-53R5

- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **AC-3** — Access Enforcement
- **IR-5** — Incident Monitoring

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control

CIS-V8

- **6.1** — Establish an Access Granting Process
- **6.8** — Define and Maintain Role-Based Access Control

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.23** — Information security for use of cloud services

NIST-CSF-2

- **DE.AE-08** — Incidents are declared when adverse events meet the defined incident criteria

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1556	Modify Authentication Process	Credential-Access
T1190	Exploit Public-Facing Application	Initial-Access

Sources

Source	URL	Tier
osv	https://osv.dev/vulnerability/GHSA-267c-6grr-h53f	T3
(consolidated)	https://osv.dev/vulnerability/GHSA-26hh-7cqf-hhc6	T3
(consolidated)	https://osv.dev/vulnerability/GHSA-36qx-fr4f-26g5	T3
CVE-2026-44575 - CVE Record	https://www.cve.org/CVERecord?id=CVE-2026-44575	T3
WAF Release - 2026-05-07 - Emergency - Cloudflare Community	https://community.cloudflare.com/t/waf-waf-release-2026-05-07-emerg...	T3
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-44575	T1

Generated 2026-05-12 06:37 UTC by TJS Security Command Center