INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-11 18:50 UTC

GHSA-c4j6-fc7j-m34r: Next.js vulnerable to server-side request forgery in applications using WebSocke

CVE VULNERABILITY | HIGH | CVSS 8.1

SCC Item ID	SCC-CVE-2026-0155
Type	CVE Vulnerability
CVE ID	CVE-2026-44578
Severity	HIGH
CVSS Base Score	8.1
Affected Products	next (npm), specific vulnerable version range not confirmed from available data
Published	2026-05-11T15:55:15Z
Discovery Source	Osv

Executive Summary

A server-side request forgery (SSRF) vulnerability in the Next.js framework allows attackers to abuse the WebSocket upgrade mechanism to force application servers into making unauthorized requests to internal or external systems. Organizations running Next.js applications with WebSocket support are exposed until the affected package is patched. If internal services such as metadata APIs, databases, or private endpoints are reachable from the Next.js server, this vulnerability can serve as an entry point for lateral movement or data exfiltration.

Technical Analysis

CVE-2026-44578 (GHSA-c4j6-fc7j-m34r) is a server-side request forgery vulnerability in the 'next' npm package, classified under CWE-918. The flaw exists in the WebSocket upgrade handling path, where insufficient validation of upgrade request parameters allows an attacker to manipulate the server-side HTTP request target. This enables requests to internal resources not intended to be externally accessible, including cloud metadata endpoints (e.g., 169.254.169.254), internal APIs, or adjacent services on private network segments. MITRE techniques associated with this vulnerability include T1090 (Proxy) and T1071.001 (Application Layer Protocol: Web Protocols). CVSS base score is reported at 8.1 (High). Affected version range and patch version details should be retrieved from osv.dev/vulnerability/GHSA-c4j6-fc7j-m34r and nvd.nist.gov/vuln/detail/CVE-2026-44578, as they are not detailed in this summary. EPSS score is not yet populated, indicating limited public exploitation data at time of writing. CISA KEV inclusion: none at this time.

Action Checklist

1. Step 1: Containment, Identify all production services running the 'next' npm package with WebSocket support enabled. Place WAF rules blocking WebSocket upgrade requests with Host or Origin headers pointing to RFC-1918 IP ranges (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16) or cloud metadata endpoints (169.254.169.254) as a temporary control. Restrict outbound server-to-server HTTP from Next.js application nodes to only explicitly required destinations via egress firewall rules.
2. Step 2: Detection, Query application and WAF logs for WebSocket upgrade requests (HTTP 101 responses) followed by unexpected HTTP requests (GET/POST) from the established socket to RFC-1918 or cloud metadata endpoints. Review Node.js process-level outbound connection logs for unexpected destinations. Search for outbound HTTP requests from internal endpoints triggered by external WebSocket upgrade activity. If using a SIEM, alert on outbound requests from Next.js server processes to 169.254.169.254 or localhost variants.
3. Step 3: Eradication, Upgrade the 'next' npm package to the patched version as identified in the GitHub advisory (github.com/advisories/GHSA-c4j6-fc7j-m34r) or NVD entry (nvd.nist.gov/vuln/detail/CVE-2026-44578). Verify package integrity via npm audit after upgrade. Remove or disable WebSocket upgrade support on any Next.js instances where it is not functionally required.
4. Step 4: Recovery, After patching, re-run npm audit to confirm no remaining high or critical findings in the dependency tree. Validate that egress controls implemented in Step 1 remain in place as a defense-in-depth measure. Monitor application logs for 24-48 hours post-patch for any anomalous outbound request patterns that may indicate prior exploitation activity.
5. Step 5: Post-Incident, Review network segmentation between Next.js application servers and internal services; SSRF exploitability depends heavily on what internal resources the application server can reach. Assess whether server-side egress filtering is part of your standard deployment baseline. If not, add it as a hardening requirement for all Node.js application deployments. Map this finding to NIST SP 800-53 control SC-7 (Boundary Protection) and SI-10 (Information Input Validation) for GRC documentation.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to senior IR leadership and legal/compliance if log review confirms any HTTP 200 responses from internal services (databases, metadata APIs, credential stores) triggered by external WebSocket upgrade requests, as this indicates successful SSRF exploitation and potential data exposure requiring breach notification assessment under applicable regulations (GDPR, HIPAA, state breach laws).
Recovery Notes	After patching the 'next' npm package to the version confirmed in GHSA-c4j6-fc7j-m34r, verify that 'npm audit' returns zero high or critical findings and that 'npm ls next' confirms no pinned transitive dependency re-introduces the vulnerable version. Maintain egress firewall rules blocking RFC-1918 and cloud metadata IP ranges on all Next.js nodes as a permanent defense-in-depth control, not just a temporary measure. Monitor application and network logs for 48 hours post-patch specifically for outbound Node.js connections to internal IP ranges, as delayed exploitation artifacts may surface if an attacker established any persistent callback mechanisms prior to remediation.

Forensic Artifacts	Web server access logs (nginx /var/log/nginx/access.log or Apache /var/log/apache2/access.log): filter for HTTP 101 Switching Protocols responses paired with Host header values resolving to RFC-1918 addresses or 169.254.169.254 — this is the primary handshake artifact unique to CVE-2026-44578 SSRF via WebSocket upgrade abuse. | Node.js process outbound socket state snapshot ('ss -tnp state established | grep node' output): captures active connections from the Next.js process to internal IP ranges at time of detection, establishing whether the SSRF reached live internal services. | npm package-lock.json (/path/to/app/package-lock.json): documents the exact vulnerable 'next' package version present during the exposure window; required for incident record and regulatory documentation of affected software versions. | tcpdump/Wireshark PCAP of outbound traffic from Next.js host (filter: dst net 10.0.0.0/8 or dst host 169.254.169.254): network-level evidence of SSRF payloads reaching internal services, including full HTTP request content if captured on the wire unencrypted. | Cloud instance metadata API response log (if applicable — output of 'curl http://169.254.169.254/latest/meta-data/' from the Next.js host): confirms whether IAM role credentials, instance identity, or environment secrets were accessible to an attacker exploiting the SSRF vector, scoping potential credential rotation requirements.

Per-Action IR Details

Step 1: Containment — Identify all production services running the 'next' npm package with WebSocket support enabled. Place WAF rules blocking malformed or unexpected WebSocket upgrade headers targeting internal IP ranges (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, 169.254.169.254) as a temporary control. Restrict outbound server-to-server HTTP from Next.js application nodes to only explicitly required destinations via egress firewall rules.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST SC-7 (Boundary Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: Run 'find /app /opt /srv -name package.json | xargs grep -l "next"' across all application nodes to enumerate Next.js deployments. For egress restriction without an enterprise firewall, apply iptables rules on each Next.js host: 'iptables -A OUTPUT -d 10.0.0.0/8 -p tcp -j DROP' (repeat for 172.16.0.0/12, 192.168.0.0/16, 169.254.169.254). Use 'ss -tnp | grep node' to identify active outbound Node.js connections before applying rules. Block WebSocket upgrade abuse at the nginx/Apache reverse proxy layer by rejecting Upgrade: websocket headers where the downstream Host value resolves to an RFC-1918 address.

Evidence: Before applying WAF or egress rules, snapshot current active Node.js outbound connections via 'ss -tnp state established | grep node > /tmp/node_connections_$(date +%s).txt'. Capture nginx or Apache access logs showing WebSocket upgrade requests ('grep -i "upgrade.*websocket" /var/log/nginx/access.log > /tmp/ws_upgrade_pre_containment.txt'). Preserve WAF logs if already in place, filtering on HTTP 101 (Switching Protocols) responses paired with RFC-1918 destination Host headers — these represent the exploit's initial handshake artifact specific to CVE-2026-44578.

Step 2: Detection — Query application and WAF logs for WebSocket upgrade requests containing Host or URL targets pointing to RFC-1918 addresses or cloud metadata IPs. Review Node.js process-level outbound connection logs for unexpected destinations. Search for HTTP 200 responses from internal endpoints triggered by external WebSocket handshake activity. If using a SIEM, alert on outbound requests from Next.js server processes to 169.254.169.254 or localhost variants.

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

Compensating: Without a SIEM, run this grep pipeline against nginx/application logs to surface SSRF-indicative WebSocket upgrade activity: 'grep -E "(upgrade|websocket|101)" /var/log/nginx/access.log | grep -E "(10\.|172\.(1[6-9]|2[0-9]|3[0-1])\.|192\.168\.|169\.254\.169\.254|localhost|127\.0\.0\.1)"'. Use tcpdump on the Next.js host's network interface to capture outbound connections from the node process: 'tcpdump -i eth0 -w /tmp/nodejs_egress_$(date +%s).pcap dst net 10.0.0.0/8 or dst host 169.254.169.254'. Deploy the free Sigma rule for SSRF detection (SigmaHQ community ruleset) converted to a grep/awk pipeline if no SIEM is available. Install osquery and run 'SELECT pid, remote_address, remote_port FROM process_open_sockets WHERE name="node" AND remote_address NOT IN (your_allowed_destinations)' on a scheduled interval.

Evidence: The specific forensic signature of CVE-2026-44578 exploitation is an HTTP 101 Switching Protocols response in the web server access log immediately followed by an outbound HTTP request from the Next.js process to an internal IP — capture both events with timestamps. Collect Node.js application-level logs (typically stdout/stderr redirected to /var/log/app/ or journald under the service unit) filtering for URL patterns including 'http://169.254.169.254', 'http://localhost', or 'http://10.' which indicate the SSRF payload reached internal services. Preserve the full HTTP request including headers for any 101 responses — the malicious Host or Forwarded header value embedded in the WebSocket upgrade is the primary attacker-controlled artifact for this CVE.

Step 3: Eradication — Upgrade the 'next' npm package to the patched version once confirmed via the GitHub advisory (github.com/advisories/GHSA-c4j6-fc7j-m34r) or NVD entry. Verify package integrity via npm audit after upgrade. Remove or disable WebSocket upgrade support on any Next.js instances where it is not functionally required.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), 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: Patch procedure for a 2-person team without automated patching: (1) 'npm outdated next' to confirm current version; (2) 'npm install next@ --save-exact' using the version confirmed in GHSA-c4j6-fc7j-m34r; (3) 'npm audit --audit-level=high' to verify the advisory no longer appears; (4) 'npm ls next' to confirm no transitive dependency pins an older vulnerable version. To disable WebSocket upgrade support on instances where it is not required, add a custom Next.js server entry point that rejects Upgrade headers: intercept the HTTP server's 'upgrade' event and call 'socket.destroy()' for all non-application WebSocket paths. Verify package integrity post-install with 'npm audit signatures' (available in npm v8.15+) to confirm registry signature validation passed.

Evidence: Before upgrading, capture 'npm ls next --json > /tmp/next_dependency_tree_pre_patch.json' to document the vulnerable version state for the incident record. Record the exact vulnerable version string from package-lock.json ('/path/to/app/package-lock.json') — the 'version' field under the 'next' entry is the legally defensible artifact confirming exposure scope. After patching, run 'npm audit --json > /tmp/npm_audit_post_patch.json' and retain both pre- and post-patch audit outputs as eradication verification evidence per NIST IR-5 (Incident Monitoring) documentation requirements.

Step 4: Recovery — After patching, re-run npm audit to confirm no remaining high or critical findings in the dependency tree. Validate that egress controls implemented in Step 1 remain in place as a defense-in-depth measure. Monitor application logs for 24-48 hours post-patch for any anomalous outbound request patterns that may indicate prior exploitation activity.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST IR-5 (Incident Monitoring), NIST SI-4 (System Monitoring), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For 24-48 hour post-patch monitoring without a SIEM, schedule a cron job on each Next.js host that runs every 15 minutes and appends anomalous egress to a watch file: '*/15 * * * * ss -tnp state established | grep node | grep -v >> /var/log/nodejs_egress_watch.log'. Set up a simple bash alerting loop: 'tail -f /var/log/nodejs_egress_watch.log | while read line; do echo "$line" | mail -s "Next.js anomalous egress" soc@yourorg.example; done'. If Wireshark/tshark is available, run a background capture filtering for outbound Node.js traffic to internal ranges: 'tshark -i eth0 -f "dst net 10.0.0.0/8 or dst host 169.254.169.254" -w /tmp/post_patch_monitor.pcap &' for the 48-hour window.

Evidence: Before returning systems to full production, collect a final 'npm audit --json' output and a snapshot of 'npm ls next --json' confirming the patched version is active — these are the recovery verification artifacts. If any 101 Switching Protocols responses or outbound requests to internal IPs were observed during the detection phase, pull and preserve the full nginx/application log segments covering the period from first suspicious WebSocket upgrade request through patch deployment, as these establish whether exploitation preceded remediation and may trigger breach notification assessment.

Step 5: Post-Incident — Review network segmentation between Next.js application servers and internal services; SSRF exploitability depends heavily on what internal resources the application server can reach. Assess whether server-side egress filtering is part of your standard deployment baseline. If not, add it as a hardening requirement for all Node.js application deployments. Map this finding to NIST SP 800-53 control SC-7 (Boundary Protection) and SI-10 (Information Input Validation) for GRC documentation.

NIST Phase: Post Incident

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

Controls: NIST SC-7 (Boundary Protection), NIST SI-10 (Information Input Validation), NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST RA-3 (Risk Assessment), CIS 4.2 (Establish and Maintain a Secure Configuration Process for Network Infrastructure), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: To assess blast radius from the CVE-2026-44578 SSRF vector without enterprise network tooling, run 'curl -s http://169.254.169.254/latest/meta-data/' from the Next.js host to confirm whether cloud instance metadata was reachable during the exposure window — document the response as a blast radius indicator. Map reachable internal services by running 'nmap -sT -p 80,443,8080,8443,5432,3306,6379,9200 10.0.0.0/24' (adjust CIDR) from the Next.js host to enumerate what an attacker with SSRF access could have reached. Use this output to scope the post-incident network segmentation review. Document findings in a lessons-learned report explicitly mapping the SSRF reachability surface to NIST SC-7 (Boundary Protection) control gaps for GRC records.

Evidence: The primary post-incident artifact for this SSRF vulnerability is a complete inventory of internal services reachable from the Next.js application server during the exposure window — document this via the nmap scan output above and any HTTP 200 responses from internal IPs observed in logs during Step 2. Collect and retain all WebSocket upgrade request logs, outbound connection snapshots, and npm audit outputs gathered during Steps 1-4 as the evidentiary record supporting the incident timeline. If cloud metadata endpoint (169.254.169.254) was reachable, assess whether IAM credentials or sensitive environment data were accessible via the metadata API and treat as a potential credential exposure incident requiring rotation per NIST IR-9 (Information Spillage Response).

Detection Guidance

Look for the following behavioral indicators: (1) WebSocket upgrade requests (HTTP 101 responses) followed immediately by outbound HTTP GET/POST requests from the application server to internal IP ranges or cloud metadata services; correlate via application-layer logs or network flow data. (2) Outbound connections from Next.js server processes to 169.254.169.254 (AWS/GCP/Azure instance metadata), 100.100.100.200 (Alibaba metadata), or localhost/127.0.0.1 variants. (3) Unexpected HTTP requests originating from the web tier to internal microservices or administrative interfaces that do not correspond to normal application traffic patterns. If using AWS CloudTrail or GCP Audit Logs, alert on instance metadata service calls not initiated by expected

automation. In WAF logs, flag upgrade requests with Host headers resolving to RFC-1918 addresses. No confirmed public IOC indicators (IPs, hashes, domains) are available for this vulnerability at this time. Monitor CISA Vulnerability Catalog (cisa.gov) and commercial threat intelligence feeds weekly for IOCs and active exploitation reports as they emerge.

Framework Mappings

MITRE-ATTACK

- **T1090** — Proxy
- **T1071.001** — Web Protocols

OWASP-TOP10-2021

- **A10:2021** — Server-Side Request Forgery (SSRF)

NIST-800-53R5

- **SC-7** — Boundary Protection
- **SI-10** — Information Input Validation

CIS-V8

- **13.4** — Perform Traffic Filtering Between Network Segments
- **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
T1090	Proxy	Command-And-Control
T1071.001	Web Protocols	Command-And-Control

Sources

Source	URL	Tier
osv	https://osv.dev/vulnerability/GHSA-c4j6-fc7j-m34r	T3
CVE-2026-44578 - CVE Details, Severity, and Analysis	https://strobes.co/vi/cve/CVE-2026-44578/	T3

Source	URL	Tier
Next.js vulnerable to server-side request forgery in ...	https://github.com/advisories/GHSA-c4j6-fc7j-m34r	T3
CVE-2026-4578 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-4578	T1
CVE Record: CVE-2026-44578	https://www.cve.org/CVERecord?id=CVE-2026-44578	T3
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-44578	T1

Generated 2026-05-11 18:50 UTC by TJS Security Command Center