INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-14 06:03 UTC

CVE-2026-33032: Nginx UI MCP Endpoint Missing Authentication Allows Complete Nginx Service Takeover

CVE VULNERABILITY | **CRITICAL** | CVSS 9.8 | **CISA KEV**

SCC Item ID	SCC-CVE-2026-0038
Type	CVE Vulnerability
CVE ID	CVE-2026-33032
Severity	CRITICAL
CVSS Base Score	9.8
EPSS Score	0.0006 (19th percentile)
KEV Status	Yes — CISA Known Exploited Vulnerability
Affected Products	nginxui nginx_ui versions 2.3.5 and prior
Published	2026-04-13T00:00:00Z
Discovery Source	Vulncheck Kev

Executive Summary

A critical unauthenticated vulnerability in Nginx UI (versions 2.3.5 and prior) allows any network attacker to take full control of Nginx web server configurations without credentials. The flaw stems from a misconfigured Model Context Protocol endpoint that bypasses authentication entirely, enabling attackers to modify, delete, or reload web server configurations at will. CISA and VulnCheck have both confirmed active exploitation, and no patch is publicly available.

Technical Analysis

CVE-2026-33032 (CVSS 9.8, CWE-306: Missing Authentication for Critical Function) affects Nginx UI versions 2.3.5 and prior. The MCP integration exposes two endpoints: /mcp (enforces IP whitelisting and AuthRequired() middleware) and /mcp_message (enforces IP whitelisting only). The critical flaw: the default IP whitelist is empty, and the middleware logic interprets an empty whitelist as 'allow all,' rendering network-level access control inoperative by default. Any unauthenticated remote attacker with network access to the /mcp_message endpoint can invoke all MCP tools, including Nginx restart (T1489), configuration file creation/modification/deletion (T1565.001), and configuration reload triggered via T1190 initial access. Exploitation requires no credentials and no prior foothold. No vendor patch was publicly available at time of

publication. Listed in both CISA KEV and VulnCheck KEV, confirming active in-the-wild exploitation. EPSS score is 0.06% (0.0006 raw), placing this in the bottom percentile of exploitability predictions; however, KEV listing supersedes EPSS as a direct indicator of active exploitation.

Action Checklist

1. Step 1: Containment. Immediately block external network access to /mcp and /mcp_message endpoints via firewall ACL, WAF rule, or reverse proxy configuration. If Nginx UI is internet-facing, take the management interface offline until patched. Identify all hosts running nginxui/nginx_ui <= 2.3.5 via asset inventory or package manager query.
2. Step 2: Detection. Review web server access logs for requests to /mcp_message from unexpected source IPs. Query SIEM for HTTP POST/GET to URI paths matching '/mcp_message' (any method, any source). Check for unexpected Nginx configuration file modifications (file integrity monitoring alerts on nginx.conf and sites-enabled/). Review Nginx UI audit logs for unauthenticated tool invocations (MCP tool names: nginx restart, config file operations). Alert on unexpected Nginx service restarts (systemctl/journald events). Correlate web log hits on /mcp_message with subsequent file system changes and process events.
3. Step 3: Eradication. No public patch is available as of publication. Apply the following interim mitigations: (1) Restrict access to Nginx UI to a dedicated management network or VPN-only VLAN; (2) Enforce a non-empty IP whitelist in Nginx UI MCP configuration to override the 'allow all' default; (3) If MCP integration is not required, disable or block the /mcp and /mcp_message routes at the reverse proxy or host firewall level. Monitor the official Nginx UI repository for patch releases.
4. Step 4: Recovery. After applying mitigations, audit all Nginx configuration files for unauthorized modifications: compare against known-good backups or version-controlled configurations. Verify Nginx service behavior is consistent with expected state. Restore any altered configs from backup. Re-enable Nginx UI access only after network controls are confirmed active and a non-empty IP whitelist is enforced. Monitor /mcp_message access logs continuously post-mitigation.
5. Step 5: Post-Incident. Conduct a controls review for all web-based management interfaces: verify authentication is enforced on every API endpoint, not just primary routes. Add automated checks for empty/permissive access control lists in management tooling. Incorporate MCP and AI-adjacent integrations into security architecture review processes, as these surfaces are increasingly exploited. Map control gaps to NIST CSF PR.AC and PR.PT control categories.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO and legal/compliance immediately if forensic analysis confirms any /mcp_message requests resulted in Nginx configuration changes (evidenced by config file modification timestamps or nginx reload events in error.log), as this constitutes confirmed exploitation with potential data exposure for any PII/PHI traffic routed through the affected Nginx instance, triggering breach notification assessment under GDPR, HIPAA, or applicable state law; also escalate if Nginx UI is found to be internet-facing with no prior access controls, given CISA-confirmed active exploitation of CVE-2026-33032.

Recovery Notes	After restoring Nginx configurations from verified backups, validate that all virtual host, upstream, and proxy_pass directives are consistent with the pre-incident baseline before resuming production traffic — pay particular attention to any new `proxy_pass` entries or Lua code blocks that could redirect traffic or exfiltrate data without visible service disruption. Monitor `/var/log/nginx/access.log` and the Nginx UI application log for `/mcp_message` access attempts continuously for a minimum of 30 days post-mitigation, as threat actors who successfully enumerated the endpoint pre-containment may return after observing that the service was briefly taken offline. Nginx UI access should not be re-enabled in any configuration until either the vendor releases a patch addressing CVE-2026-33032 or a non-empty IP whitelist restricting MCP access to a dedicated management VLAN is confirmed active and tested.
Forensic Artifacts	Nginx UI application access log (path configured in app.ini, e.g. /etc/nginx-ui/app.log or configured log_dir): filter for POST/GET requests to /mcp and /mcp_message without Authorization headers or Bearer tokens — unauthenticated tool invocations are the primary indicator of CVE-2026-33032 exploitation and will appear as MCP tool calls (e.g., nginx reload, config write) with no preceding authentication event. | File system modification timestamps on /etc/nginx/nginx.conf, /etc/nginx/sites-enabled/*, and /etc/nginx/conf.d/*: use `stat` or `find /etc/nginx -newer /var/log/nginx/access.log -type f` to identify config files modified during the exploitation window — this directly evidences whether an attacker used the MCP endpoint to alter Nginx routing, add reverse proxies, or inject malicious upstream definitions. | Nginx error log (/var/log/nginx/error.log): grep for '[notice] signal process started' and 'reload' events outside of scheduled maintenance windows — each successful Nginx reload following an unauthorized /mcp_message request confirms that attacker-supplied configuration changes were applied to the live production service. | Nginx UI app.ini or equivalent configuration file: capture the MCP access control section to document whether the IP whitelist was empty (the vulnerable default) at the time of exploitation — this establishes the root cause configuration state and is required evidence for both internal RCA and any regulatory breach notification documentation. | Network flow logs or Nginx access logs for downstream traffic anomalies following any detected config modification: if an attacker added a malicious proxy_pass directive, look for new upstream destinations in access logs (`awk '{print $7}' /var/log/nginx/access.log | sort | uniq -c | sort -rn`) and correlate with netflow or `ss -tnp` output to identify any data exfiltration channels introduced via the unauthorized Nginx configuration changes.

Per-Action IR Details

Step 1: Containment — Immediately block external network access to /mcp and /mcp_message endpoints via firewall ACL, WAF rule, or reverse proxy configuration. If Nginx UI is internet-facing, take the management interface offline until patched. Identify all hosts running nginxui/nginx_ui <= 2.3.5 via asset inventory or package manager query.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: isolate affected systems to prevent further exploitation of the unauthenticated MCP endpoint while preserving operational continuity of downstream Nginx services where possible.

Controls: NIST IR-4 (Incident Handling), NIST SC-7 (Boundary Protection), NIST CM-7 (Least Functionality), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory)

Compensating: On each Linux host running Nginx UI, immediately insert a host-level block using iptables: `iptables -I INPUT -p tcp --dport -j DROP` and persist with `iptables-save`. For URI-level blocking without a WAF, add a Nginx location block in the upstream reverse proxy: `location ~* ^/mcp(_message)?$ { deny all; return 403; }` then reload with

`nginx -s reload`. Enumerate all affected hosts via: `dpkg -l | grep nginx-ui` (Debian/Ubuntu) or `rpm -qa | grep nginx-ui` (RHEL/CentOS) or `find / -name 'nginx-ui' -type f 2>/dev/null` cross-referenced against your asset inventory CSV.

Evidence: Before blocking, capture the current state of active connections to the Nginx UI management port using `ss -tnp | grep ` and `netstat -antp | grep ` — document any established sessions from unexpected source IPs as potential active exploitation. Record the current Nginx UI process tree with `ps auxf | grep nginx-ui` and note the parent PID. Export firewall state before modification: `iptables -L -n -v > iptables_precontainment_$(date +%Y%m%d%H%M%S).txt`. Take a timestamped snapshot of /etc/nginx/nginx.conf and /etc/nginx/sites-enabled/* as the pre-containment baseline.

Step 2: Detection — Review web server access logs for requests to /mcp_message from unexpected source IPs. Query SIEM for HTTP POST/GET to URI paths matching '/mcp_message' (any method, any source). Check for unexpected Nginx configuration file modifications (file integrity monitoring alerts on nginx.conf and sites-enabled/). Review Nginx UI audit logs for unauthenticated tool invocations. Correlate with any unexpected Nginx service restarts.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: identify indicators of exploitation against the /mcp_message endpoint, distinguish scanning/reconnaissance from active config manipulation, and establish timeline of unauthorized Nginx UI MCP tool invocations.

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-2 (Event Logging), NIST AU-3 (Content of Audit Records), NIST SI-4 (System Monitoring), NIST IR-5 (Incident Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: Without a SIEM, grep Nginx UI's access log directly: `grep -E '/(mcp|mcp_message)' /var/log/nginx/access.log | awk '{print $1, $7, $9}' | sort | uniq -c | sort -rn` to surface source IPs, URI paths, and HTTP status codes. For file integrity monitoring without an enterprise tool, use AIDE (`aide --check`) or a manual sha256 baseline: `find /etc/nginx -type f -exec sha256sum {} \; > /tmp/nginx_baseline_$(date +%Y%m%d).txt` then diff against a prior snapshot. Monitor Nginx UI's own audit log at the path configured in its `app.ini` (typically under the Nginx UI data directory, e.g. `/etc/nginx-ui/app.log` or the configured `log_dir`). For unauthenticated invocation detection, grep for MCP tool call entries without a preceding authentication token: `grep -i 'mcp\|tool_call\|invoke' /path/to/nginx-ui/app.log | grep -iv 'auth\|token\|bearer'`.

Evidence: Collect the following before log rotation or system changes: (1) Full Nginx UI access log (`/var/log/nginx/access.log` or the path defined in nginx-ui's configuration) covering at minimum 30 days prior — filter for `/mcp` and `/mcp_message` URI patterns including any URL-encoded variants (`%2Fmcp`, `%2Fmcp_message`); (2) Nginx UI application log showing MCP tool invocations — specifically look for `nginx reload`, `config write`, `config delete`, or equivalent tool names called without authentication context; (3) `/var/log/nginx/error.log` for config reload events (`nginx: [notice] signal process started`) timestamped outside of normal maintenance windows; (4) File system modification timestamps on `/etc/nginx/nginx.conf`, `/etc/nginx/sites-enabled/*`, and `/etc/nginx/conf.d/*` using `stat` or `ls -la --full-time`; (5) OS-level auth logs (`/var/log/auth.log` or `/var/log/secure`) for any SSH or sudo activity coinciding with detected MCP requests, indicating potential follow-on access after config manipulation.

Step 3: Eradication — No public patch is available as of publication. Apply the following interim mitigations: (1) Restrict access to Nginx UI to a dedicated management network or VPN-only VLAN; (2) Enforce a non-empty IP whitelist in Nginx UI MCP configuration to override the 'allow all' default; (3) If MCP integration is not required, disable or block the /mcp and /mcp_message routes at the reverse proxy or host firewall level. Monitor the Nginx UI GitHub repository (https://github.com/0xJacky/nginx-ui) for patch releases.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication: in the absence of a vendor patch for CVE-2026-33032, eradication consists of closing the unauthenticated attack path by enforcing network-layer and application-layer access controls on the MCP endpoint, and verifying no malicious persistence was introduced via unauthorized Nginx config changes during any exploitation window.

Controls: NIST SI-2 (Flaw Remediation), NIST CM-7 (Least Functionality), NIST SC-7 (Boundary Protection), NIST SI-5 (Security Alerts, Advisories, and Directives), CIS 7.1 (Establish and Maintain a Vulnerability Management

Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 4.2 (Establish and Maintain a Secure Configuration Process for Network Infrastructure)

Compensating: To enforce the Nginx UI MCP IP whitelist without enterprise tooling, edit the Nginx UI `app.ini` configuration file and set the `[mcpAllowedIP]` or equivalent MCP access control field to a specific management IP range (e.g., `10.10.1.0/24`); restart Nginx UI and verify the setting is active by attempting a curl request from a non-whitelisted IP: `curl -X POST http://:/mcp_message` — a 403 or connection refused confirms the whitelist is enforced. If disabling MCP entirely, block at host firewall and additionally confirm no active MCP client is configured by reviewing Nginx UI's `app.ini` for any `mcp_enabled = true` or equivalent directive. Subscribe to GitHub release notifications for https://github.com/0xJacky/nginx-ui via RSS feed or `watch` tooling as a zero-cost patch availability monitor.

Evidence: Before finalizing eradication controls, capture: (1) The current Nginx UI `app.ini` (or equivalent config file) to document the pre-mitigation MCP access control state — specifically whether the IP whitelist was empty/unset, confirming the 'allow all' default described in CVE-2026-33032; (2) A full diff of all Nginx configuration files against version-controlled or backup baselines (`diff /etc/nginx/nginx.conf /backup/nginx.conf`) to identify any unauthorized server blocks, proxy_pass directives, or upstream definitions added via MCP tool calls; (3) Check for webshells or unauthorized includes injected into Nginx config: `grep -r 'include\|lua_code_cache\|content_by_lua\|access_by_lua' /etc/nginx/` as these are attacker-favored persistence mechanisms via Nginx config manipulation; (4) Record the Nginx UI version string (`nginx-ui --version` or from package metadata) to confirm the vulnerable version for documentation and regulatory reporting purposes.

Step 4: Recovery — After applying mitigations, audit all Nginx configuration files for unauthorized modifications: compare against known-good backups or version-controlled configurations. Verify Nginx service behavior is consistent with expected state. Restore any altered configs from backup. Re-enable Nginx UI access only after network controls are confirmed active and a non-empty IP whitelist is enforced. Monitor /mcp_message access logs continuously post-mitigation.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery: restore Nginx service to a verified clean state by validating all configuration files against known-good baselines, confirming no malicious routing or proxy rules were introduced via the unauthenticated MCP endpoint, and re-enabling Nginx UI only under enforced network access controls.

Controls: NIST IR-4 (Incident Handling), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CP-10 (System Recovery and Reconstitution), NIST AU-11 (Audit Record Retention), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 7.3 (Perform Automated Operating System Patch Management)

Compensating: Perform a three-way config audit without enterprise tooling: (1) `nginx -t` to validate current config syntax; (2) `diff -rq /etc/nginx/ /path/to/git/repo/nginx/` or against a tarball backup to surface unauthorized file additions or modifications; (3) `md5sum /etc/nginx/nginx.conf` compared against a pre-incident hash recorded in your baseline. For continuous post-mitigation monitoring of `/mcp_message` without a SIEM, deploy a lightweight tail-based alert: `tail -F /var/log/nginx/access.log | grep --line-buffered '/mcp_message' | while read line; do echo "$(date): ALERT - MCP access detected: $line" >> /var/log/mcp_alert.log; done &` — run as a systemd service or screen session. Verify Nginx service integrity by checking the binary hash: `sha256sum $(which nginx)` against the vendor-published hash for the installed version.

Evidence: Before restoring from backup, preserve the potentially compromised configuration state as forensic evidence: `tar czf /forensics/nginx_config_compromised_$(date +%Y%m%d%H%M%S).tar.gz /etc/nginx/` — this is critical if regulatory breach notification is required, as it documents the scope of any config tampering. Capture `nginx -V` output to record compile-time modules and flags, as attackers who had MCP access could have modified Lua modules or dynamic module configurations. Review `/var/log/nginx/error.log` for any `[emerg]` or `[crit]` entries coinciding with the exploitation window, which would indicate config reload attempts — successful reloads confirm attacker-modified configs were live in production. Document the restoration timestamp and the git commit hash or backup timestamp of the known-good config used for recovery.

Step 5: Post-Incident — Conduct a controls review for all web-based management interfaces: verify authentication is enforced on every API endpoint, not just primary routes. Add automated checks for empty/permissive access control lists in management tooling. Incorporate MCP and AI-adjacent integrations

into security architecture review processes, as these surfaces are increasingly exploited. Map control gaps to NIST CSF PR.AC and PR.PT control categories.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: conduct lessons-learned analysis focused on the systemic gap exposed by CVE-2026-33032 — specifically, the absence of authentication enforcement on MCP/AI-integration endpoints — and update security architecture review processes to treat MCP surfaces as a new attack class requiring explicit coverage.

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST CA-7 (Continuous Monitoring), NIST RA-3 (Risk Assessment), NIST AC-3 (Access Enforcement), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 5.1 (Establish and Maintain an Inventory of Accounts)

Compensating: Without enterprise GRC tooling, conduct the management interface audit using a structured checklist: enumerate all web-based admin panels (Nginx UI, Grafana, Portainer, etc.) via `ss -tlnp` on each server, then for each, manually test for unauthenticated API endpoint access using `curl -v -X POST http://:/api/` without credentials — document any 200-series responses as findings. For automated ACL permissiveness checks, write a cron-executed script that greps Nginx UI `app.ini` and similar config files for empty whitelist fields and alerts via email or syslog. For MCP/AI integration inventory, add a standing checklist item to your change management process: any new software integration must declare MCP, REST, or LLM tool-call endpoints and document their authentication mechanism before deployment approval.

Evidence: For the lessons-learned record and any regulatory documentation, compile: (1) The full timeline of detected `/mcp_message` requests from access logs with source IPs, timestamps, and HTTP methods — this establishes the exploitation window for breach scope determination; (2) A list of all Nginx configuration files that were modified or accessed during the incident window, with before/after diffs and restoration timestamps; (3) Documentation of the Nginx UI version deployed and the date it was last reviewed against vendor advisories, to support root cause analysis of the detection gap; (4) The network topology showing how Nginx UI was exposed (internet-facing vs. internal) at the time of exploitation, as this determines blast radius and may trigger regulatory notification obligations if PII/PHI was routed through the affected Nginx instance; (5) Output of the post-incident management interface audit, documenting all other web admin panels reviewed and their authentication enforcement status.

Detection Guidance

Primary detection surface: web access logs on hosts running Nginx UI. Query for HTTP requests to '/mcp_message' from any source IP, particularly POST requests with MCP tool invocation payloads (e.g., nginx restart, config file write/delete operations). File integrity monitoring (FIM) alerts on Nginx configuration directories (/etc/nginx/, /usr/local/nginx/conf/, or custom paths) are a strong secondary indicator; unauthorized file writes or deletes in these paths following an /mcp_message request indicate active exploitation. Alert on unexpected Nginx service restarts (systemd/journald events for nginx.service stop/start outside change windows). If a SIEM or EDR is deployed, correlate web log hits on /mcp_message with subsequent file system changes and process events for the nginx or nginx-ui service account. No public IOC hashes or IPs are confirmed at this time; behavioral detection is the primary approach.

Framework Mappings

MITRE-ATTACK

- **T1565.001** — Stored Data Manipulation
- **T1190** — Exploit Public-Facing Application
- **T1489** — Service Stop

NIST-800-53R5

- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-6** — Configuration Settings
- **SI-4** — System Monitoring
- **IA-2** — Identification and Authentication (Organizational Users)
- **IR-5** — Incident Monitoring

OWASP-TOP10-2021

- **A07:2021** — Identification and Authentication Failures

CIS-V8

- **6.3** — Require MFA for Externally-Exposed Applications
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

HIPAA-SECURITY

- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures

ISO-27001-2022

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

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
T1565.001	Stored Data Manipulation	Impact
T1190	Exploit Public-Facing Application	Initial-Access
T1489	Service Stop	Impact

Sources

Source	URL	Tier
vulncheck_kev	https://nvd.nist.gov/vuln/detail/CVE-2026-33032	T1
CVE-2026-33032, nginx-ui's Unauthenticated MCP Endpoint Allows ...	https://www.endorlabs.com/vulnerability/cve-2026-33032	T3
CVE-2026-33032 - Exploits & Severity - Feedly	https://feedly.com/cve/CVE-2026-33032	T3
CVE-2026-33032 CPEs | Tenable®	https://www.tenable.com/cve/CVE-2026-33032/cpes	T3
CVE-2026-33032 - CRITICAL Vulnerability | SOC Defenders	https://socdefenders.ai/cves/CVE-2026-33032	T3
CISA KEV	https://www.cisa.gov/known-exploited-vulnerabilities-catalog	T1

Generated 2026-04-14 06:03 UTC by TJS Security Command Center