INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-17 07:18 UTC

CVE-2026-54410: nanoMODBUS through v1.23.0 contains an off-by-one buffer overflow in the recv_msg_header() function ...

CVE VULNERABILITY | HIGH | CVSS 8.6

SCC Item ID	SCC-CVE-2026-0315
Type	CVE Vulnerability
CVE ID	CVE-2026-54410
Severity	HIGH
CVSS Base Score	8.6
EPSS Score	0.0054 (41th percentile)
Affected Products	nanoMODBUS through v1.23.0
Published	2026-06-14T18:17:20.330
Discovery Source	Nvd

Executive Summary

CVE-2026-54410 is a high-severity off-by-one buffer overflow in nanoMODBUS, an open-source Modbus/TCP library widely used in industrial control systems, embedded devices, and OT environments. A remote, unauthenticated attacker can crash affected devices or, on bare-metal and RTOS targets without memory protection, read or write device register values without authentication. Organizations running nanoMODBUS v1.23.0 or earlier in operational technology or industrial environments should treat this as a priority remediation item.

Technical Analysis

CVE-2026-54410 affects nanoMODBUS through v1.23.0. The vulnerability resides in the recv_msg_header() function of the Modbus/TCP server implementation. An attacker sends a crafted MBAP (Modbus Application Protocol) frame with the Length field set to 255, causing the library to write one attacker-controlled byte one position past the end of a 260-byte receive buffer (CWE-193: Off-by-One Error; CWE-787: Out-of-Bounds Write). The overflow corrupts the adjacent buffer-index field in the nanoMODBUS state structure. On platforms with memory protection (Linux, etc.), this reliably causes denial of service via invalid memory access. On bare-metal microcontrollers and RTOS targets lacking MPU/MMU enforcement, the corrupted index also enables one-byte information disclosure and unintended writes to Modbus register addresses via the Write

Multiple Registers (FC16) handler path. No authentication is required. Attack vector is network-accessible with low complexity. MITRE ATT&CK mappings: T1499 (Endpoint Denial of Service), T1190 (Exploit Public-Facing Application). CVSS base score: 8.6. EPSS: 0.541% (41st percentile). Not currently listed on CISA KEV. Monitor the nanoMODBUS GitHub repository or equivalent official source for a patched release addressing this vulnerability; availability and timeline are subject to vendor discretion.

Action Checklist

1. Step 1: Containment, Identify all systems in your environment running nanoMODBUS v1.23.0 or earlier. Inventory embedded devices, PLCs, RTUs, and RTOS-based controllers that include this library. Immediately restrict network access to Modbus/TCP port 502 on affected devices using firewall ACLs or network segmentation, allowing only known engineering workstations and SCADA servers to communicate with these devices. On bare-metal or RTOS targets without memory protection, treat remote access as highest priority for isolation (per NIST AC-4, Information Flow Enforcement).
2. Step 2: Detection, Query asset inventory and software bill of materials (SBOM) records for nanoMODBUS as a dependency (CIS 2.1, Establish and Maintain a Software Inventory). In network logs, look for inbound Modbus/TCP sessions to port 502 where the MBAP Length field equals 255 (0xFF); this is the specific malformed frame pattern described in CVE-2026-54410. Alert on unexpected process crashes or watchdog resets on embedded targets. On RTOS devices, log anomalous FC16 (Write Multiple Registers) transactions to register addresses outside expected operational ranges (per NIST AU-2, Event Logging; NIST AU-6, Audit Record Review, Analysis, and Reporting).
3. Step 3: Eradication, Monitor the nanoMODBUS GitHub repository for a patched release addressing CVE-2026-54410. When available, update all affected devices and rebuild any firmware or software packages that embed the library. If a patch is not yet available, implement input validation at the network boundary: deploy an IDS/IPS rule to drop MBAP frames with Length field = 255 before they reach the target device. Document any exceptions with formal risk acceptance (per NIST SI-4, System Monitoring; CIS 7.1, Establish and Maintain a Vulnerability Management Process).
4. Step 4: Recovery, After patching or applying mitigations, verify the nanoMODBUS version in deployed firmware matches the remediated release. On bare-metal and RTOS targets, confirm memory protection units (MPU) are configured and enforced where hardware supports it. Restore normal Modbus/TCP access only after confirming firewall rules are in place. Monitor for anomalous Modbus register write activity (FC16) in the 72 hours post-remediation to confirm no prior exploitation occurred (per NIST AU-6, Audit Record Review, Analysis, and Reporting).
5. Step 5: Post-Incident, This CVE exposes a control gap in OT/embedded software supply chain visibility. Verify your organization maintains an accurate SBOM for all firmware and embedded software, including third-party libraries such as nanoMODBUS (CIS 2.1). Establish a process to receive security advisories from embedded library maintainers. Review network segmentation controls separating Modbus/TCP devices from general IT networks (NIST AC-4; CIS 4.4, Implement and Manage a Firewall on Servers). Where bare-metal targets lack MPU enforcement, document this as an accepted architectural risk and prioritize hardware refresh cycles that include memory protection capabilities (CIS 7.2, Establish and Maintain a Remediation Process).

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to OT/ICS security leadership and notify asset owners if network monitoring detects any inbound Modbus/TCP frame with MBAP Length=0xFF (0b11111111) reaching an unpatched nanoMODBUS device, or if post-incident register-state comparison reveals FC16 writes to process control registers outside established operational baselines — either condition indicates likely exploitation of CVE-2026-54410 in a production OT environment, which may trigger ICS incident reporting obligations under CISA reporting guidelines or sector-specific regulations (NERC CIP for electric, TSA directives for pipeline).
Recovery Notes	After patching or applying compensating controls, restore Modbus/TCP access incrementally — re-enable communication to one device zone at a time and validate register integrity via engineering workstation polling before proceeding to the next. Run continuous tshark capture filtered to FC16 transactions on port 502 for a minimum of 72 hours post-remediation, comparing register write activity against the pre-incident operational baseline to detect any persistence of unauthorized writes that may have occurred before containment. On bare-metal and RTOS targets confirmed to lack MPU enforcement, document these as elevated-residual-risk assets and flag them for priority hardware refresh in the next capital planning cycle.
Forensic Artifacts	Network pcap from OT segment showing Modbus/TCP sessions to port 502 with MBAP Length field = 0xFF (255) — the specific malformed frame that triggers the off-by-one overflow in recv_msg_header(); preserve full packet bytes including MBAP transaction ID and unit ID for attribution | SCADA historian or data acquisition logs recording all FC16 (Write Multiple Registers) transactions, including source IP, timestamp, target register address range, and written values — compared against pre-incident operational register baseline to identify unauthorized writes | Device crash/reset logs from embedded targets: watchdog timeout events, RTOS exception handler output, or serial console fault messages indicating stack corruption consistent with the nanoMODBUS buffer overflow; on Linux-based embedded hosts, collect /proc/kmsg and any core dumps before reimaging | Firmware binary artifacts (pre- and post-patch) with SHA-256 hashes and embedded nanoMODBUS version strings extracted via `strings` or `binwalk` — establishes definitive evidence of which library version was running at time of incident and confirms successful remediation | ARP tables, switch MAC address tables, and Modbus/TCP session logs identifying all source IPs that sent frames to port 502 on affected devices during the exposure window — used to determine whether exploitation attempts originated from inside the OT network (pivot from IT) or directly from external access

Per-Action IR Details

Step 1: Containment — Identify all systems in your environment running nanoMODBUS v1.23.0 or earlier. Inventory embedded devices, PLCs, RTUs, and RTOS-based controllers that include this library. Immediately restrict network access to Modbus/TCP port 502 on affected devices using firewall ACLs or network segmentation, allowing only known engineering workstations and SCADA servers to communicate with these devices. On bare-metal or RTOS targets without memory protection, treat remote access as highest priority for isolation (per NIST AC-4 — Information Flow Enforcement).

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.2 (Establish and Maintain a Secure Configuration Process for Network Infrastructure)

Compensating: On a 2-person team without enterprise NAC: immediately push ACL rules on the upstream switch or router blocking all inbound TCP/502 except from whitelisted engineering workstation and SCADA server IPs —

`iptables -I INPUT -p tcp --dport 502 ! -s -j DROP` on any Linux-based gateway. For Cisco IOS border switches use `ip access-list extended BLOCK_MODBUS / deny tcp any host eq 502 / permit ip any any`. Capture a Wireshark pcap on the OT network segment for at least 15 minutes before applying ACLs to preserve pre-containment traffic baselines.

Evidence: Before applying any firewall ACL or network isolation, capture: (1) full-packet pcap of the OT/ICS network segment showing all active Modbus/TCP sessions to port 502 — preserve MBAP headers to identify any in-flight Length=255 (0xFF) frames; (2) ARP table and switch MAC address tables (`show mac address-table` / `arp -a`) to map IP-to-device associations for all port-502 talkers; (3) current process list and memory state snapshots on any Linux-based embedded hosts (`ps aux`, `/proc//maps`) before isolation alters live state. On bare-metal/RTOS targets without OS-level capture capability, log all current Modbus register states via a read-only engineering workstation poll before isolating.

Step 2: Detection — Query asset inventory and software bill of materials (SBOM) records for nanoMODBUS as a dependency (CIS 2.1 — Establish and Maintain a Software Inventory). In network logs, look for inbound Modbus/TCP sessions to port 502 where the MBAP Length field equals 255 (0xFF); this is the specific malformed frame pattern described in CVE-2026-54410. Alert on unexpected process crashes or watchdog resets on embedded targets. On RTOS devices, log anomalous FC16 (Write Multiple Registers) transactions to register addresses outside expected operational ranges (per NIST AU-2 — Event Logging; NIST AU-6 — Audit Record Review, Analysis, and Reporting).

NIST Phase: Detection Analysis

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

Controls: CIS 2.1 (Establish and Maintain a Software Inventory), NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting)

Compensating: Without a SIEM, use Wireshark/tshark with the filter `tcp.port == 502 && modbus.len == 255` against a pcap or live OT tap to identify malformed MBAP frames characteristic of CVE-2026-54410 exploitation attempts. For SBOM discovery on Linux-based embedded builds, run `grep -r 'nanoMODBUS\|nanomodbus' /path/to/firmware/rootfs` or scan binary artifacts with `strings firmware.bin | grep -i nanomodbus`. For crash/watchdog detection on RTOS targets without syslog, correlate device uptime counters via SCADA historian reads — an unexpected counter reset indicates a probable crash event. Write a Sigma rule detecting MBAP Length=0xFF on port 502 deployable to any log shipper.

Evidence: Before and during detection activity, preserve: (1) raw network pcaps from the OT segment capturing the full MBAP header bytes of all port-502 sessions — the off-by-one in recv_msg_header() is triggered specifically when MBAP Length byte = 0xFF (255), so preserve framing intact; (2) SCADA historian or data acquisition logs showing FC16 write transactions with timestamps, originating IP, and target register ranges for comparison against operational baselines; (3) device syslog or serial console output from embedded targets showing any stack fault messages, watchdog timeout events, or exception vectors — on RTOS targets these may appear as reboot events or fault handler logs in non-volatile flash memory; (4) SBOM and firmware build manifests identifying nanoMODBUS version strings linked in firmware images.

Step 3: Eradication — Monitor the nanoMODBUS GitHub repository for a patched release addressing CVE-2026-54410. When available, update all affected devices and rebuild any firmware or software packages that embed the library. If a patch is not yet available, implement input validation at the network boundary: deploy an IDS/IPS rule to drop MBAP frames with Length field = 255 before they reach the target device. Document any exceptions with formal risk acceptance (per NIST SI-4 — System Monitoring; CIS 7.1 — Establish and Maintain a Vulnerability Management Process).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Without enterprise patch management: manually rebuild firmware images embedding nanoMODBUS after updating to the patched library release, then verify the corrected version string is present in the output binary

(`strings firmware_new.bin | grep nanomodbus`). As an interim boundary control before a patch is available, deploy a Snort/Suricata rule targeting the CVE-2026-54410 malformed MBAP frame: `alert tcp any any -> $OT_SUBNET 502 (msg:"CVE-2026-54410 nanoMODBUS MBAP Length=0xFF"; content:"|FF|"; offset:4; depth:1; sid:2026544100; rev:1;)` — set rule action to `drop` in inline IPS mode. A 2-person team can apply this to a pfSense or Suricata instance sitting inline on the OT segment.

Evidence: Before applying any firmware update or IDS/IPS rule that alters packet flow, capture: (1) pre-patch firmware binary hashes (SHA-256) of all affected device firmware images for change-control verification; (2) current Modbus register state via an authenticated engineering workstation poll — a pre-exploitation register baseline is needed to detect whether unauthorized FC16 writes altered register values prior to eradication; (3) any volatile memory forensics available on Linux-based embedded targets (core dumps, `/var/log/` contents, `/proc/kmsg` kernel ring buffer) before the patch reimage clears live state. On bare-metal targets, extract non-volatile flash logs if the device supports it before reflashing.

Step 4: Recovery — After patching or applying mitigations, verify the nanoMODBUS version in deployed firmware matches the remediated release. On bare-metal and RTOS targets, confirm memory protection units (MPU) are configured and enforced where hardware supports it. Restore normal Modbus/TCP access only after confirming firewall rules are in place. Monitor for anomalous Modbus register write activity (FC16) in the 72 hours post-remediation to confirm no prior exploitation occurred (per NIST AU-6; D3-LAM — Local Account Monitoring for any associated session anomalies).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 7.3 (Perform Automated Operating System Patch Management)

Compensating: Without EDR for OT: use a Wireshark/tshark continuous capture on the OT segment filtered to `tcp.port == 502 && modbus.func_code == 16` (FC16 Write Multiple Registers) for the 72-hour post-remediation window, writing rolling pcaps to disk (`tshark -i eth0 -f 'tcp port 502' -b filesize:102400 -w /captures/modbus_postpatch.pcap`). Compare FC16 target register addresses and values against the pre-incident operational baseline — deviations may indicate prior unauthorized writes that persisted through the patch cycle. Confirm patched firmware version by extracting and grepping the deployed image.

Evidence: Before restoring normal Modbus/TCP access (i.e., lifting firewall ACLs), capture: (1) a post-patch firmware binary hash to confirm the remediated nanoMODBUS version is present and matches expected build artifacts; (2) a full Modbus register state read across all process data objects on affected PLCs/RTUs, compared against the pre-incident baseline — any discrepancy in FC16-writable registers (holding registers, coils) may indicate exploitation occurred before containment; (3) device uptime and reset counters at the moment of access restoration as a baseline for ongoing watchdog monitoring. Retain pre- and post-patch pcaps for comparison during the 72-hour monitoring window.

Step 5: Post-Incident — This CVE exposes a control gap in OT/embedded software supply chain visibility. Verify your organization maintains an accurate SBOM for all firmware and embedded software, including third-party libraries such as nanoMODBUS (CIS 2.1). Establish a process to receive security advisories from embedded library maintainers. Review network segmentation controls separating Modbus/TCP devices from general IT networks (NIST AC-4; CIS 4.4 — Implement and Manage a Firewall on Servers). Where bare-metal targets lack MPU enforcement, document this as an accepted architectural risk and prioritize hardware refresh cycles that include memory protection capabilities (CIS 7.2 — Establish and Maintain a Remediation Process).

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity (Lessons Learned)

Controls: CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), NIST AC-4 (Information Flow Enforcement), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For a 2-person team lacking a commercial SBOM platform: build a firmware dependency registry in a shared spreadsheet or Git repository, with columns for library name, version, CVE watch status, and device assignments — update it as part of every firmware build process. Subscribe to the nanoMODBUS GitHub repository's release and security advisory notifications (free) and to CISA ICS-CERT advisories (https://www.cisa.gov/ics-advisories) for OT-specific CVE disclosures. Use `grep` or `binwalk` in a CI pipeline step to detect nanoMODBUS version strings in firmware artifacts before they are deployed.

Evidence: Post-incident documentation should preserve: (1) the final incident timeline showing when CVE-2026-54410 was first detectable in network logs versus when containment was applied — this gap quantifies dwell-time exposure for the lessons-learned report; (2) the complete list of devices found running nanoMODBUS v1.23.0 or earlier, their network zones, and their MPU capability status — this becomes the architectural risk register entry for bare-metal targets; (3) all pcaps, register-state snapshots, and firmware hashes collected during the incident, retained per your AU-11 retention policy for potential regulatory or insurance reporting needs.

Detection Guidance

Primary detection surface is network traffic inspection on Modbus/TCP port 502. Write an IDS/IPS signature to flag or block MBAP frames where the Length field (bytes 5-6 of the MBAP header, big-endian) equals 0x00FF (255). This is the specific malformed frame that triggers the off-by-one write. Secondary detection: monitor device health logs on embedded targets for unexpected resets, watchdog triggers, or crash dumps, these are behavioral indicators of exploitation attempts on platforms without memory protection. For RTOS devices, alert on FC16 (Write Multiple Registers) transactions targeting register addresses outside defined operational ranges; corrupted buffer-index state may cause the FC16 handler to write to unintended addresses. If SBOM tooling is in place, scan for nanoMODBUS as a dependency in firmware packages and flag all instances at or below v1.23.0 for prioritized review. Per NIST AU-2 and AU-6, ensure Modbus/TCP session logs are retained and reviewed for anomalous Length field values. No public IOCs (IPs, domains, hashes) are available at this time; exploitation is opportunistic based on Modbus/TCP service exposure rather than actor-attributed infrastructure.

Framework Mappings

MITRE-ATTACK

- **T1499** — Endpoint Denial of Service
- **T1190** — Exploit Public-Facing Application

NIST-800-53R5

- **SC-5** — Denial-of-Service Protection
- **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
- **SI-16** — Memory Protection

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures

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
T1190	Exploit Public-Facing Application	Initial-Access

Sources

Source	URL	Tier
nvd	https://nvd.nist.gov/vuln/detail/CVE-2026-54410	T1
CVE-2026-54410: nanoMODBUS Off-by-One Buffer Overflow in ...	https://threat-modeling.com/cve-2026-54410-nanomodbus-modbus-tcp-bu.. .	T3
CVE-2026-54410 - Vulnerability Database - Cyber Defence	https://www.cyber-defence.io/tools/cve/CVE-2026-54410	T3
CVE-2026-54410 Off-by-One Buffer Overflow in nanoMODBUS ...	https://x.com/VulmonFeeds/status/2066231712879714500	T3
CVE-2026-54410 - nanoMODBUS TCP Server Off-by-One Buffer ...	https://www.enigma-global.com/cve/8375951	T3

Generated 2026-06-17 07:18 UTC by TJS Security Command Center