INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-25 06:50 UTC

glibc scanf %mc Off-by-One Heap Buffer Overflow (CVE-2026-5450)

CVE VULNERABILITY | CRITICAL | CVSS 9.8

SCC Item ID	SCC-CVE-2026-0077
Type	CVE Vulnerability
CVE ID	CVE-2026-5450
Severity	CRITICAL
CVSS Base Score	9.8
EPSS Score	0.0004 (11th percentile)
Affected Products	Microsoft Azure Linux 3.0, azl3 glibc 2.38-19
Published	2026-04-25T01:05:58
Discovery Source	Msrc Patch Tuesday

Executive Summary

A critical heap buffer overflow in the glibc C library affects Microsoft Azure Linux 3.0, carrying a CVSS score of 9.8. The flaw exists in how glibc processes a specific string format specifier and can allow an attacker to execute arbitrary code or escalate privileges on affected systems. Organizations running Azure Linux 3.0 workloads should treat this as a priority patching event; Microsoft disclosed the vulnerability as part of the April 2026 Patch Tuesday cycle.

Technical Analysis

CVE-2026-5450 is an off-by-one heap buffer overflow (CWE-122, CWE-193) in the glibc scanf implementation, specifically triggered by the %mc format specifier. The error allows a write one byte beyond the allocated heap buffer boundary. Affected package: azl3 glibc 2.38-19 on Microsoft Azure Linux 3.0. CVSS base score: 9.8 (Critical); EPSS score: 0.038% (11th percentile, no observed exploitation in the wild as of disclosure). Not listed on CISA KEV. Exploitation could enable arbitrary code execution (MITRE T1203) or local privilege escalation (MITRE T1068) depending on heap layout at runtime. Successful exploitation typically requires the ability to supply a crafted input string to a process using scanf with %mc. Patch disclosed via MSRC April 2026 Patch Tuesday. Sources: MSRC Update Guide (https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-5450), NVD (https://nvd.nist.gov/vuln/detail/CVE-2026-5450).

Action Checklist

1. Step 1: Containment, Identify all Azure Linux 3.0 hosts running glibc 2.38-19 (package query: rpm -q glibc on azl3 nodes). Restrict network exposure of affected workloads where possible while patching is staged.
2. Step 2: Detection, Query your asset inventory and CMDB for Azure Linux 3.0 instances. On Linux hosts, run 'rpm -q glibc' to confirm the 2.38-19 version string. Review process audit logs (auditd, syslog) for anomalous heap allocation patterns or unexpected process crashes (SIGSEGV, SIGABRT) in applications using scanf.
3. Step 3: Eradication, Apply the updated glibc package for azl3 as released by Microsoft via the Azure Linux 3.0 package feed. Consult the MSRC advisory at https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-5450 for the specific patched package version. Verify that Microsoft's Azure Linux 3.0 package repository has published the patched glibc version before running 'tdnf update glibc' on azl3 systems. Check the MSRC advisory for the specific patched version number.
4. Step 4: Recovery, After patching, verify the updated glibc version is installed on all affected hosts. Restart any long-running services that loaded the affected glibc at startup to ensure the patched library is in use. Monitor application stability post-patch and review logs for any crash recurrence.
5. Step 5: Post-Incident, Audit your Azure Linux 3.0 fleet for patch coverage gaps. Evaluate whether your asset inventory accurately captured all azl3 workloads. Review patching SLAs for Critical-rated CVEs in OS-level libraries and adjust if response time exceeded policy thresholds.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO and legal/compliance immediately if any pre-patch SIGSEGV or SIGABRT crash logs on internet-facing azl3 workloads cannot be ruled out as exploitation attempts, or if those workloads process PII, PHI, or regulated data that would trigger breach notification obligations under GDPR, HIPAA, or applicable state law.
Recovery Notes	Patch verification must confirm both the RPM version string and the SHA-256 hash of '/lib64/libc.so.6' match the MSRC-published patched build — installing the correct package version is necessary but not sufficient if a compromised host had the glibc binary replaced in place. Any azl3 host that exhibited SIGSEGV or SIGABRT in a scanf-invoking process during the exposure window should be treated as potentially compromised and undergo full process memory and filesystem integrity review before being returned to production. Monitor all patched azl3 nodes for a minimum of 72 hours post-restart, specifically watching for new crash events, unexpected child process spawning from glibc-linked daemons (consistent with post-exploitation persistence), and any new outbound connections from previously well-behaved services.

Forensic Artifacts	Core dump files in '/var/crash/' or the path specified in '/proc/sys/kernel/core_pattern' from processes using scanf with %mc format strings — heap metadata in these dumps will show the off-by-one corruption pattern specific to CVE-2026-5450 and can confirm whether a crash was organic or exploit-driven. | '/var/log/audit/audit.log' ANOM_ABEND records correlating SIGSEGV or SIGABRT signals to specific PID and process names on azl3 nodes during the window between April 2026 Patch Tuesday disclosure and confirmed patch deployment — these are the primary evidence source for potential exploitation attempts. | Output of 'grep "libc-2.38" /proc/*/maps' captured before service restarts — any process retaining a memory map to the vulnerable glibc binary after the patch was installed indicates it was not restarted and may have been running exploitable code continuously. | 'lsof +L1' output showing deleted-but-still-open file handles to the old glibc shared library — this artifact confirms which services were running against the unpatched library at the time of eradication and is essential for determining the full exposure window per service. | Pre- and post-patch SHA-256 hashes of '/lib64/libc.so.6' from each azl3 host — if a post-patch hash does not match the MSRC-published value for the patched glibc build, it indicates either a failed patch application or potential binary tampering on that specific node.

Per-Action IR Details

Step 1: Containment — Identify all Azure Linux 3.0 hosts running glibc 2.38-19 (package query: rpm -q glibc on azl3 nodes). Restrict network exposure of affected workloads where possible while patching is staged.

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)

Compensating: Run the following one-liner across all azl3 nodes via SSH or Azure Run Command to enumerate affected hosts: 'for host in $(cat azl3_hostlist.txt); do ssh $host "rpm -q glibc | grep 2.38-19 && echo $host VULNERABLE"; done > glibc_exposure.txt'. Use Azure NSG rules or host-based iptables to block inbound access to services that call scanf with user-controlled format strings (e.g., custom daemons, FTP services). If network segmentation is not feasible, disable non-essential internet-facing services on affected nodes until the patch is staged.

Evidence: Before isolating any node, capture: (1) the full running process list ('ps auxf > /tmp/procs_$(hostname)_$(date +%F).txt') to identify which applications were actively using glibc scanf at the time of containment; (2) open network connections ('ss -tulnp > /tmp/netstat_$(hostname)_$(date +%F).txt') to identify which services were externally reachable and therefore exploitable; (3) a memory snapshot if any process is exhibiting active heap corruption symptoms (SIGSEGV/SIGABRT) using 'gcore ' before the process is restarted or killed — this preserves heap state that could confirm in-flight exploitation of CVE-2026-5450.

Step 2: Detection — Query your asset inventory and CMDB for Azure Linux 3.0 instances. On Linux hosts, run 'rpm -q glibc' to confirm the 2.38-19 version string. Review process audit logs (auditd, syslog, /var/log/messages) for anomalous heap allocation patterns or unexpected process crashes (SIGSEGV, SIGABRT) in applications using scanf.

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), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 8.2 (Collect Audit Logs)

Compensating: Use auditd with a rule targeting execve calls from processes that link against glibc's scanf family: 'auditctl -a always,exit -F arch=b64 -S execve -k glibc_scanf_watch'. Query existing auditd logs for SIGSEGV and SIGABRT signals against known scanf-using daemons: 'ausearch -m ANOM_ABEND -ts today | grep -E "vsftpd|sshd|bash|custom_daemon"'. For crash detection without a SIEM, run 'journalctl -p 3 --since "7 days ago" | grep

-E "segfault|SIGABRT|core dumped" > crash_review.txt' across all azl3 nodes. Use osquery with the query 'SELECT name, path, pid FROM processes JOIN process_open_files USING (pid) WHERE path LIKE "%libc%"' to enumerate live processes with glibc loaded.

Evidence: Preserve before analysis: (1) '/var/log/audit/audit.log' and '/var/log/messages' from all azl3 nodes covering the 30 days prior to detection — the off-by-one heap overflow in scanf's %mc specifier may produce repeated SIGSEGV or SIGABRT entries against the same process if an attacker was probing; (2) application-specific crash logs and core dump files in '/var/crash/' or configured core dump paths (check 'cat /proc/sys/kernel/core_pattern') — core dumps from processes using scanf with %mc will show heap metadata corruption consistent with CVE-2026-5450; (3) '/var/log/secure' or 'journalctl _COMM=sshd' to identify any privilege escalation attempts co-incident with crash events.

Step 3: Eradication — Apply the updated glibc package for azl3 as released by Microsoft via the Azure Linux 3.0 package feed. Consult the MSRC advisory at https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-5450 for the specific patched package version. Use your standard patching pipeline (tdnf update glibc on azl3 systems).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST CM-3 (Configuration Change Control), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: For teams without an enterprise patch management platform, execute the following across azl3 nodes via Azure Run Command or SSH in a scripted loop: 'tdnf update glibc -y && rpm -q glibc >> /tmp/patch_verification.txt'. Verify the patched package version string matches the MSRC advisory before marking a host clean. If tdnf cannot reach the Azure Linux 3.0 package feed from a network-isolated node, download the updated glibc RPM directly from the azl3 package mirror, validate its SHA-256 checksum against the published value in the MSRC advisory, and install with 'rpm -Uvh glibc-.azl3.x86_64.rpm'. Do not patch and immediately resume services — proceed to the evidence capture in the recovery step first.

Evidence: Before applying the patch, snapshot the current glibc binary and its linked dependencies: 'rpm -qi glibc > /tmp/glibc_preimage_$(hostname).txt && sha256sum /lib64/libc.so.6 >> /tmp/glibc_preimage_$(hostname).txt'. This establishes a forensic baseline of the vulnerable binary on each node, which is required if any host is later suspected of pre-patch compromise. Also capture 'lsof | grep libc' output to identify all processes holding open file handles to the vulnerable library — this list drives the mandatory service restart step in recovery.

Step 4: Recovery — After patching, verify the updated glibc version is installed on all affected hosts. Restart any long-running services that loaded the affected glibc at startup to ensure the patched library is in use. Monitor application stability post-patch and review logs for any crash recurrence.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, And Information Integrity), NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Verify patched glibc is in active use — not just installed — by confirming no process retains a memory-mapped reference to the old vulnerable library: 'grep "libc-2.38" /proc/*/maps 2>/dev/null'. Any PID returned is still running against the unpatched library and must be restarted. Use 'needs-restarting' (part of yum-utils, available on azl3) or manually check 'lsof +L1' for deleted library mappings. Post-restart, monitor '/var/log/messages' and 'journalctl -f' for 48 hours for any new SIGSEGV or SIGABRT events in applications that previously used scanf with %mc format strings — recurrence after patching could indicate a separate exploit or a service that failed to restart cleanly.

Evidence: After patch and service restarts, collect: (1) 'rpm -qa --last | head -20' to confirm glibc package replacement timestamp on each host; (2) 'sha256sum /lib64/libc.so.6' compared against the expected hash published in the MSRC advisory or the azl3 package feed — this confirms the correct patched binary is on disk; (3) a 48-hour window of '/var/log/audit/audit.log' and 'journalctl' output post-restart, retained as evidence that no crash recurrence occurred after eradication — this log set is your proof-of-clean for any compliance or regulatory reporting obligation.

Step 5: Post-Incident — Audit your Azure Linux 3.0 fleet for patch coverage gaps. Evaluate whether your asset inventory accurately captured all azl3 workloads. Review patching SLAs for Critical-rated CVEs in OS-level libraries and adjust if response time exceeded policy thresholds.

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-5 (Security Alerts, Advisories, And Directives), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Conduct a reconciliation between your CMDB/asset inventory and the actual output of 'rpm -q glibc' collected during Step 1 — any azl3 host that appeared in the rpm sweep but not in your CMDB is an inventory gap that must be remediated. Build or update an osquery scheduled query to continuously report glibc version across azl3 nodes: 'SELECT version FROM deb_packages WHERE name="glibc"' (adapt for RPM: 'SELECT version FROM rpm_packages WHERE name="glibc"'). Document the elapsed time from Microsoft's April 2026 Patch Tuesday disclosure of CVE-2026-5450 to confirmed patch deployment across 100% of azl3 fleet; if this exceeded your Critical SLA (typically 15 days for CVSS 9.x), formally update the SLA and the patching runbook.

Evidence: Retain for lessons-learned and potential audit: (1) the complete glibc_exposure.txt enumeration from Step 1 showing all affected hosts and their patch status at time of discovery; (2) patch verification outputs ('rpm -qi glibc' post-patch) from all azl3 nodes as evidence of remediation completeness; (3) any crash logs (SIGSEGV/SIGABRT from Steps 2 and 4) that occurred prior to patching, which represent potential exploitation attempts against CVE-2026-5450 and should be preserved for 90 days minimum per NIST AU-11 (Audit Record Retention) or applicable regulatory retention requirements.

Detection Guidance

Run 'rpm -q glibc' on all Azure Linux 3.0 hosts; output of '2.38-19' confirms exposure. In auditd logs, look for repeated SIGSEGV or SIGABRT signals from processes using scanf-based input parsing. No public proof-of-concept exploit or IOC signatures were published as of the April 2026 disclosure. Check security forums and exploit databases (ExploitDB, GitHub) for updates. EPSS score (0.038%) indicates exploitation is not currently observed in the wild; focus detection effort on confirming affected package presence rather than active exploitation indicators. Cross-reference your CMDB and cloud asset inventory against known azl3 deployments in your environment.

Framework Mappings

MITRE-ATTACK

- **T1068** — Exploitation for Privilege Escalation
- **T1203** — Exploitation for Client Execution

NIST-800-53R5

- **AC-6** — Least Privilege
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring

CIS-V8

- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management
- **5.4** — Restrict Administrator Privileges to Dedicated Administrator Accounts

ISO-27001-2022

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

SOC2-TSC

- **CC6.3** — Authorizes, modifies, or removes access

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1068	Exploitation for Privilege Escalation	Privilege-Escalation
T1203	Exploitation for Client Execution	Execution

Sources

Source	URL	Tier
MSRC Update Guide	https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-5450	T1
(consolidated)	https://api.msrc.microsoft.com/cvrf/v3.0/cvrf/2026-Apr	T1
CVE-2026-5450 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-5450	T1
CVE-2026-5450 - Exploits & Severity - Feedly	https://feedly.com/cve/CVE-2026-5450	T3
Linux Distros Unpatched Vulnerability : CVE-2026-5450 | Tenable®	https://www.tenable.com/plugins/nessus/308167	T3

Generated 2026-04-25 06:50 UTC by TJS Security Command Center