INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-05 08:19 UTC

Critical Out-of-Bounds Write in Linux ksmbd SMB2 EA Handling (CVE-2026-31705)

CVE VULNERABILITY | CRITICAL | CVSS 9.8

SCC Item ID	SCC-CVE-2026-0122
Type	CVE Vulnerability
CVE ID	CVE-2026-31705
Severity	CRITICAL
CVSS Base Score	9.8
EPSS Score	0.0005 (14th percentile)
Affected Products	Microsoft Azure Linux 3.0, azl3 kernel 6.6.134.1-2; Linux ksmbd subsystem (SMB2 server implementation)
Published	2026-05-05T01:03:22
Discovery Source	Msrc Patch Tuesday

Executive Summary

A critical vulnerability in the Linux kernel's ksmbd SMB file-sharing module allows a remote attacker to execute code or corrupt kernel memory without authentication. Microsoft Azure Linux 3.0 is confirmed affected, with broader Linux distribution exposure flagged by Red Hat and Tenable. Organizations running Azure Linux 3.0 workloads with SMB services accessible over the network face immediate risk of server compromise.

Technical Analysis

CVE-2026-31705 is a critical out-of-bounds write (CWE-787) in the Linux kernel's ksmbd module, specifically within the smb2_get_ea() function. The flaw occurs during Extended Attribute (EA) alignment processing in SMB2 request handling; a misalignment calculation allows a write operation to exceed its allocated buffer boundary, enabling kernel memory corruption or remote code execution. CVSS base score: 9.8 (AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H). No authentication is required; attack complexity is low. EPSS score: 0.046% (13.9th percentile) as of disclosure; exploitation has not been observed in the wild per available data, but the vulnerability profile is consistent with high-value exploitation targets. Affected package: azl3 kernel 6.6.134.1-2 on Microsoft Azure Linux 3.0. The ksmbd subsystem was introduced in Linux 5.15 as an in-kernel SMB3 server; any distribution shipping ksmbd with an unpatched kernel version should be considered potentially exposed. Disclosed as part of Microsoft's May 2026 Patch Tuesday cycle. MITRE ATT&CK mapping:

T1210 (Exploitation of Remote Services), T1133 (External Remote Services). No public exploit code confirmed at time of item generation. CISA KEV: not listed.

Action Checklist

1. **Containment:** Identify all Azure Linux 3.0 systems (azl3 kernel 6.6.134.1-2) with ksmbd enabled and SMB ports (TCP 445, TCP 139) accessible from untrusted networks. Immediately restrict inbound SMB access via network ACLs or security groups to trusted hosts only. If SMB access is not required, disable the ksmbd module: 'rmmod ksmbd' (runtime) and blacklist it in /etc/modprobe.d/ to prevent reload on reboot.
2. **Detection:** Audit active ksmbd deployments: run 'lsmod | grep ksmbd' on all Linux hosts. Query cloud asset inventory for Azure Linux 3.0 instances. Review network flow logs for inbound TCP 445 traffic from unexpected sources. Monitor kernel logs (/var/log/kern.log or journalctl -k) for anomalous SMB2 EA processing errors or kernel oops/BUG entries referencing smb2_get_ea. No public IOC signatures exist for pre-exploit reconnaissance specific to this CVE; treat any unexpected SMB2 negotiation from external IPs as an indicator worth reviewing.
3. **Eradication:** Apply the updated azl3 kernel package from the Microsoft Azure Linux 3.0 security advisory (reference: MSRC CVE-2026-31705 update guide at https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-31705). Run 'dnf update kernel' on Azure Linux 3.0 systems and reboot to load the patched kernel. Confirm the patched version is active post-reboot with 'uname -r'. For other Linux distributions running ksmbd, monitor Red Hat (https://access.redhat.com/security/cve/cve-2026-31705) and upstream kernel security advisories for distribution-specific patches.
4. **Recovery:** After patching, confirm ksmbd is running the patched kernel version and that SMB2 EA requests are processed without error. Restore any temporarily restricted SMB access only after patch verification. Re-enable network monitoring for TCP 445 traffic and set alerting thresholds for anomalous SMB connection volumes. Review cloud security group rules to ensure SMB exposure is limited to the minimum required set of source IPs.
5. **Post-Incident:** This vulnerability exposes a control gap in in-kernel service exposure: ksmbd enabled on internet-adjacent systems without compensating network controls. Implement a standing policy requiring all in-kernel network service modules (ksmbd, nbd, etc.) to be explicitly approved and documented before enablement. Add ksmbd module status to your asset hardening baseline checks.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, legal, and cloud operations leadership immediately if kernel oops or BUG entries referencing 'smb2_get_ea' are found in kern.log on any host, if SMB connections from external IPs to Azure Linux 3.0 hosts are confirmed during the vulnerability window (indicating possible pre-patch exploitation), or if any Azure Linux 3.0 host running ksmbd also stores, processes, or transmits regulated data (PII, PHI, PCI-DSS cardholder data), as unauthenticated kernel-level remote code execution against such a host triggers breach notification assessment obligations.

Recovery Notes	After kernel patch deployment and reboot, confirm the active kernel version via 'uname -r' on every previously affected host and cross-reference against the patched package version in the MSRC CVE-2026-31705 advisory before restoring any SMB network access. Monitor 'journalctl -k' for ksmbd-related kernel errors for a minimum of 72 hours post-restoration, with particular attention to any BUG or oops entries referencing smb2_get_ea or adjacent EA-handling functions, which would indicate either incomplete patching or a secondary exploitation attempt. Maintain reduced SMB network exposure — ACL-restricted to approved source IPs only — as a permanent hardening outcome of this incident, and verify no Azure NSG rules revert to broader TCP 445 ingress during the 30-day post-incident monitoring window.
Forensic Artifacts	Kernel ring buffer logs (dmesg / journalctl -k) — a successful or attempted exploit of CVE-2026-31705 via the smb2_get_ea out-of-bounds write would produce kernel BUG, general protection fault, or Oops entries in the kernel log referencing the ksmbd subsystem; absence of such entries does not rule out exploitation if the attacker achieved stable kernel code execution without crashing. | SMB2 packet captures on TCP 445 (and 139) — raw pcap from tcpdump or cloud packet mirroring showing SMB2 SESSION_SETUP, TREE_CONNECT, and subsequent QUERY_INFO or SET_INFO commands with extended attribute payloads; a CVE-2026-31705 exploit attempt would appear as an SMB2 EA request with a malformed or oversized attribute buffer from an unauthenticated or newly authenticated source IP. | Azure NSG / VPC flow logs for inbound TCP 445 and 139 — source IP, destination IP, bytes transferred, and session duration for all SMB connections in the 7-day pre-containment window; novel source IPs, unusually short session durations (probe/scan pattern), or large inbound byte counts from a single source (exploit payload delivery) are the key indicators to isolate. | ksmbd service and share configuration files (/etc/ksmbd/ksmbd.conf, /etc/samba/smb.conf) — documents which SMB shares, authentication modes (null sessions permitted?), and EA handling options were active at the time of exposure; null authentication or guest access enabled would lower the exploitation bar for CVE-2026-31705 from network-adjacent to fully unauthenticated remote. | Volatile memory image (LiME or avml capture) taken before rmmod ksmbd and before reboot — if pre-patch exploitation is suspected, a kernel-resident implant or modified kernel function table (rootkit) installed via the CVE-2026-31705 kernel code execution primitive exists only in memory and will be destroyed on reboot; this artifact is the only forensic evidence of post-exploitation persistence at the kernel level.

Per-Action IR Details

Containment — Identify all Azure Linux 3.0 systems (azl3 kernel 6.6.134.1-2) with ksmbd enabled and SMB ports (TCP 445, TCP 139) accessible from untrusted networks. Immediately restrict inbound SMB access via network ACLs or security groups to trusted hosts only. If SMB access is not required, disable the ksmbd module: 'rmmod ksmbd' (runtime) and blacklist it in /etc/modprobe.d/ to prevent reload on reboot.

NIST Phase: Containment

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

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)

Compensating: For teams without centralized firewall management: run 'iptables -I INPUT -p tcp --dport 445 -j DROP && iptables -I INPUT -p tcp --dport 139 -j DROP' on each affected host immediately, then persist with 'iptables-save > /etc/iptables/rules.v4'. Use a bash loop over your host inventory: 'for h in $(cat azl3_hosts.txt); do ssh $h "lsmod | grep ksmbd && echo $h EXPOSED"; done'. Blacklist ksmbd via 'echo "blacklist ksmbd" >> /etc/modprobe.d/ksmbd-block.conf' and run 'dracut --force' to rebuild initramfs so the block persists across reboots. Confirm module is unloaded with 'lsmod | grep ksmbd' returning empty.

Evidence: Before isolating or restricting network access, capture the following: (1) Current ksmbd connection table — run 'ss -tnp | grep :445' and 'ss -tnp | grep :139' to record any active SMB sessions including remote IPs and PIDs at time of containment. (2) ksmbd process and module state — 'lsmod | grep ksmbd', 'ps aux | grep ksmbd', and 'cat /proc/net/tcp' filtered to port 0x01BD (445) and 0x008B (139) in hex. (3) Network flow logs from Azure NSG or VPC flow logs showing inbound TCP 445/139 connections in the 72 hours prior to containment — specifically any source IPs that are not in the approved SMB consumer list. (4) Kernel ring buffer snapshot — 'dmesg --time-format iso > /evidence/dmesg_pre_containment.txt' to preserve any existing ksmbd oops or BUG traces referencing 'smb2_get_ea' before system state changes. (5) Memory image if compromise is suspected — use 'avml' or 'LiME' kernel module to capture a full memory dump prior to rmmod, as in-kernel exploit artifacts exist only in volatile memory.

Detection — Audit active ksmbd deployments: run 'lsmod | grep ksmbd' on all Linux hosts. Query cloud asset inventory for Azure Linux 3.0 instances. Review network flow logs for inbound TCP 445 traffic from unexpected sources. Monitor kernel logs (/var/log/kern.log or journalctl -k) for anomalous SMB2 EA processing errors or kernel oops/BUG entries referencing smb2_get_ea. No public IOC signatures exist for pre-exploit reconnaissance specific to this CVE; treat any unexpected SMB2 negotiation from external IPs as an indicator worth reviewing.

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-2 (Event Logging), NIST IR-5 (Incident Monitoring), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without a SIEM, deploy a targeted detection script: 'journalctl -k --since "72 hours ago" | grep -E "(ksmbd|smb2_get_ea|general protection fault|BUG:|kernel BUG|Oops)" > /evidence/ksmbd_kernel_events.txt'. For network-level detection, run tcpdump on each exposed host: 'tcpdump -i eth0 -w /evidence/smb_capture.pcap tcp port 445 or tcp port 139' and analyze with Wireshark filtering on 'smb2 && smb2.cmd == 0x0b' (IOCTL) and extended attribute requests (SMB2 QUERY_INFO with InfoType 0x02). Write a Sigma rule targeting syslog for ksmbd kernel oops: condition on 'ksmbd' AND ('BUG' OR 'general protection fault' OR 'smb2_get_ea') in kern.log. Use osquery with 'SELECT * FROM kernel_modules WHERE name = "ksmbd";' scheduled every 15 minutes across the fleet via osqueryd.

Evidence: Collect before any system changes: (1) Full kernel log export — 'journalctl -k --since "7 days ago" -o json > /evidence/kernel_journal_7d.json' — search for strings 'smb2_get_ea', 'ksmbd', 'out-of-bounds', 'BUG:', 'Oops', and 'general protection fault' which would indicate in-kernel memory corruption triggered by malformed SMB2 EA requests. (2) Azure NSG flow logs or equivalent — filter for inbound TCP 445 and 139 from source IPs outside the approved SMB consumer CIDR ranges; timestamps of first connection from any novel source IP are critical for establishing attacker dwell time. (3) ksmbd service logs if verbose logging is enabled — '/var/log/ksmbd.log' or equivalent as configured in ksmbd.conf; look for malformed SMB2 QUERY_INFO or SET_INFO requests with oversized or malformed EA (extended attribute) payloads. (4) tcpdump packet capture of SMB2 traffic — capture SMB2 session setup and subsequent EA request commands; a CVE-2026-31705 exploit attempt would manifest as an SMB2 SET_INFO or QUERY_INFO request with a crafted EA buffer designed to trigger the out-of-bounds write in smb2_get_ea. (5) Cloud asset inventory export — query Azure Resource Graph for all VMs running Azure Linux 3.0 (azl3) with TCP 445 exposed in NSG rules, cross-referenced against the kernel version 6.6.134.1-2 from instance metadata.

Eradication — Apply the updated azl3 kernel package from the Microsoft Azure Linux 3.0 security advisory (reference: MSRC CVE-2026-31705 update guide at https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-31705). Run 'dnf update kernel' on Azure Linux 3.0 systems and reboot to load the patched kernel. Confirm the patched version is active post-reboot with 'uname -r'. For other Linux distributions running ksmbd, monitor Red Hat (https://access.redhat.com/security/cve/cve-2026-31705) and upstream kernel security advisories for distribution-specific patches.

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-7 (Least Functionality), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For teams without automated patch orchestration: stage the patch on a test Azure Linux 3.0 VM first — run 'dnf update kernel --downloadonly' to cache the package, then 'rpm -Vp kernel-.rpm' to verify the RPM signature against Microsoft's GPG key before deployment. Automate rollout with a simple bash script iterating over the host list: 'for h in $(cat azl3_hosts.txt); do ssh $h "dnf update kernel -y && reboot"; done'. Post-reboot, validate the patched kernel is active: 'for h in $(cat azl3_hosts.txt); do ssh $h "uname -r"; done' and confirm no host is still on 6.6.134.1-2. If a host cannot be rebooted immediately, maintain the ksmbd blacklist and firewall block as interim compensating controls and document the exception with a risk acceptance timestamp. Note: the URLs provided in the advisory step are unverified by this session — validate that https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-31705 resolves to the correct MSRC advisory before following patch instructions.

Evidence: Before applying the kernel patch and rebooting: (1) Capture the pre-patch kernel version and loaded module state — 'uname -a > /evidence/pre_patch_kernel.txt && lsmod > /evidence/pre_patch_modules.txt' — to establish a before/after comparison baseline. (2) If compromise is suspected on any host, take a full memory image using LiME ('insmod lime.ko path=/evidence/mem.lime format=lime') prior to reboot, since a successful CVE-2026-31705 exploit achieving kernel code execution may have installed a kernel-resident backdoor or rootkit that will not survive reboot but must be captured for forensic analysis. (3) Export the running ksmbd configuration — 'cat /etc/ksmbd/ksmbd.conf' and 'cat /etc/samba/smb.conf' if applicable — to document which shares and users were exposed at the time of the vulnerability window. (4) Collect RPM package verification output — 'rpm -Va kernel' — before patching to detect any tampering with kernel files that would indicate the system was compromised prior to this remediation action.

Recovery — After patching, confirm ksmbd is running the patched kernel version and that SMB2 EA requests are processed without error. Restore any temporarily restricted SMB access only after patch verification. Re-enable network monitoring for TCP 445 traffic and set alerting thresholds for anomalous SMB connection volumes. Review cloud security group rules to ensure SMB exposure is limited to the minimum required set of source IPs.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SC-7 (Boundary Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Validate SMB2 EA processing on the patched kernel without a commercial monitoring tool: craft a benign SMB2 extended attribute test using 'smbclient' — 'smbclient ///testshare -U testuser -c "setea testfile.txt TestAttr TestValue"' — and confirm it completes without triggering any kernel BUG or oops entries in 'journalctl -k'. Set up a lightweight connection volume alert using a cron job: 'watch -n 60 "ss -s | grep ESTAB"' with a threshold script that emails the team if established SMB connections exceed baseline. Use 'fail2ban' with a custom filter on '/var/log/kern.log' matching 'ksmbd' error patterns to auto-block source IPs generating anomalous SMB EA errors post-patch. Re-audit Azure NSG rules using the Azure CLI: 'az network nsg rule list --nsg-name --resource-group --output table' to verify TCP 445 ingress is restricted to approved source IP ranges only.

Evidence: Post-patch verification artifacts to capture and retain: (1) Confirmed patched kernel version — 'uname -r > /evidence/post_patch_kernel.txt' — to document remediation completion for audit trail per NIST AU-11 (Audit Record Retention). (2) ksmbd functional test results — output of SMB2 EA test commands as described in compensating controls — confirming the specific code path triggering CVE-2026-31705 (smb2_get_ea out-of-bounds write) no longer produces errors. (3) Kernel log snapshot post-reboot — 'journalctl -k --since "reboot" > /evidence/kernel_post_patch.txt' — confirming absence of ksmbd-related BUG or oops entries after service restoration. (4) NSG rule export post-review — document the final approved inbound TCP 445 source IP list as evidence of attack surface reduction applied during this incident.

Post-Incident — This vulnerability exposes a control gap in in-kernel service exposure: ksmbd enabled on internet-adjacent systems without compensating network controls. Implement a standing policy requiring all in-kernel network service modules (ksmbd, nbd, etc.) to be explicitly approved and documented before enablement. Add ksmbd module status to your asset hardening baseline checks. Review whether SMB is required on the affected systems or whether it can be permanently disabled as an attack surface reduction measure.

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 CM-7 (Least Functionality), NIST SI-5 (Security Alerts, Advisories, and Directives), NIST RA-3 (Risk Assessment), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: For teams without a formal CMDB or configuration management platform: create a ksmbd-specific hardening check script — 'lsmod | grep ksmbd && echo FAIL || echo PASS' — and add it to a weekly cron-driven compliance scan across all Linux hosts, outputting results to a shared log. Use an osquery scheduled query ('SELECT * FROM kernel_modules WHERE name = "ksmbd";') to provide continuous visibility into ksmbd load state across the fleet. Formalize the in-kernel module approval policy as a one-page standard operating procedure referencing this incident as the justification, mapping to CIS 2.3 (Address Unauthorized Software) for audit purposes. Submit the ksmbd exposure finding to your vulnerability management backlog with a risk rating informed by this CVE's CVSS 9.8 score and the unauthenticated remote exploitation vector.

Evidence: Artifacts required for lessons-learned documentation and audit closure: (1) Incident timeline reconstruction — compile pre-containment dmesg captures, NSG flow log exports, and patch verification records into a chronological incident record per NIST IR-5 (Incident Monitoring) requirements. (2) Scope confirmation artifact — final list of all Azure Linux 3.0 hosts that had ksmbd loaded and TCP 445 exposed, with confirmation that each is now patched to the remediated kernel version, documenting the full blast radius of CVE-2026-31705 in this environment. (3) Control gap documentation — written record of how ksmbd came to be enabled on internet-adjacent systems without network ACL controls, identifying the configuration management failure that allowed azl3 kernel 6.6.134.1-2 with ksmbd active to be deployed without a firewall boundary — input for updating the hardening baseline. (4) Policy update record — the approved in-kernel network service module policy or baseline check addition, version-controlled, referencing CVE-2026-31705 as the driving incident, retained for the next compliance audit cycle.

Detection Guidance

Primary detection method is inventory-based: enumerate all Linux hosts running ksmbd with 'lsmod | grep ksmbd' and cross-reference against the affected kernel version (azl3 kernel 6.6.134.1-2). In cloud environments, query Azure Resource Graph or equivalent asset inventory for Azure Linux 3.0 instances. Network-level: scan for TCP 445 listeners on Linux hosts and flag any with external exposure. Log-based: review kernel ring buffer (journalctl -k --since) for oops, BUG, or memory fault entries referencing ksmbd or smb2_get_ea, these may indicate exploitation attempts or successful memory corruption. No confirmed public exploit signatures or network-based detection rules specific to CVE-2026-31705 are available as of disclosure. IDS/IPS rules for malformed SMB2 EA requests may provide partial coverage; consult your vendor's signature update channel for CVE-2026-31705-specific rules. EPSS score of 0.046% indicates low observed exploitation probability at time of disclosure; this should be re-evaluated as signatures mature.

Framework Mappings

MITRE-ATTACK

- **T1133** — External Remote Services
- **T1210** — Exploitation of Remote Services

NIST-800-53R5

- **AC-17** — Remote Access
- **AC-20** — Use of External Systems
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **SC-7** — Boundary Protection
- **AC-6** — Least Privilege
- **SI-2** — Flaw Remediation
- **SI-16** — Memory Protection

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **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
- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1133	External Remote Services	Persistence
T1210	Exploitation of Remote Services	Lateral-Movement

Sources

Source	URL	Tier
MSRC Update Guide	https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-31705	T1
(consolidated)	https://api.msrc.microsoft.com/cvrf/v3.0/cvrf/2026-May	T1
CVE-2026-31705 - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-31705	T1
CVE-2026-31705 - Red Hat Customer Portal	https://access.redhat.com/security/cve/cve-2026-31705	T3

Source	URL	Tier
Linux Distros Unpatched Vulnerability : CVE-2026-31705 | Tenable®	https://www.tenable.com/plugins/nessus/311670	**T3**

Generated 2026-05-05 08:19 UTC by TJS Security Command Center