INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-27 06:39 UTC

CVE-2026-39834: Infinite Loop DoS in golang.org/x/crypto/ssh Affecting Microsoft Azure Linux cert-manager

CVE VULNERABILITY | CRITICAL | CVSS 9.1

SCC Item ID	SCC-CVE-2026-0227
Type	CVE Vulnerability
CVE ID	CVE-2026-39834
Severity	CRITICAL
CVSS Base Score	9.1
EPSS Score	0.0004 (13th percentile)
Affected Products	Microsoft azl3 cert-manager 1.12.15-6 on Azure Linux 3.0; golang.org/x/crypto/ssh (upstream)
Published	2026-05-27T01:40:27
Discovery Source	Msrc Patch Tuesday

Executive Summary

A critical-severity denial-of-service vulnerability (CVE-2026-39834) in the Go extended cryptography SSH library allows an unauthenticated attacker to trigger an infinite loop by sending oversized SSH channel writes, crashing affected services. Microsoft's cert-manager component for Azure Linux 3.0 (version 1.12.15-6) is confirmed affected, posing availability risk to Kubernetes certificate automation pipelines in Azure cloud environments. Organizations running cert-manager on Azure Linux 3.0 should treat this as a priority patching item given the critical CVSS score of 9.1, even though active exploitation has not been publicly confirmed.

Technical Analysis

CVE-2026-39834 is a CWE-835 (Loop with Unreachable Exit Condition) flaw in golang.org/x/crypto/ssh. An attacker can trigger an uncontrolled resource consumption condition by sending large channel writes over an SSH connection, causing the affected process to enter an infinite loop and become unresponsive. This maps to MITRE ATT&CK T1499.004 (Endpoint Denial of Service: Application or System Exploitation). CVSS base score is 9.1 (Critical); EPSS score is 0.00042 (12.7th percentile, indicating low current exploitation probability relative to other published vulnerabilities). The vulnerability is not listed in CISA's Known Exploited Vulnerabilities catalog. Confirmed affected component: Microsoft azl3 cert-manager 1.12.15-6 on Azure Linux 3.0. The

upstream golang.org/x/crypto/ssh package is also affected. cert-manager automates TLS certificate lifecycle management in Kubernetes clusters, meaning exploitation could disrupt certificate issuance and renewal, causing cascading TLS failures across workloads. Disclosed as part of Microsoft Patch Tuesday May 2026. Source authority: MSRC (T1), NVD (T1).

Action Checklist

1. Step 1: Containment, Identify all Azure Linux 3.0 hosts running azl3 cert-manager 1.12.15-6. Restrict SSH access to cert-manager nodes using Azure Network Security Groups (NSGs), application security groups (ASGs), and host-level firewall rules. Limit inbound SSH to known management IP ranges only. Reference: Azure Well-Architected Framework (Security pillar), CIS Azure Foundations 7.1 (Ensure Network Security Group Flow Log Retention is Greater than 90 Days).
2. Step 2: Detection, Query your asset inventory for Azure Linux 3.0 nodes with cert-manager installed. Run 'rpm -q azl3-cert-manager' or equivalent to confirm version 1.12.15-6 presence. Review process monitoring and system logs for cert-manager processes exhibiting high CPU utilization or hung states (indicative of infinite loop). Check SIEM for SSH connection anomalies: unusually large channel write payloads, repeated connection attempts to cert-manager endpoints, or process restarts. Reference: NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 8.2 (Collect Audit Logs), D3-SFA (System File Analysis), D3-LAM (Local Account Monitoring).
3. Step 3: Eradication, Apply the updated azl3 cert-manager package from the Microsoft Azure Linux 3.0 security repository as released under the May 2026 Patch Tuesday advisory (MSRC: CVE-2026-39834). For upstream Go projects using golang.org/x/crypto/ssh directly, update the dependency to the patched version identified in the upstream advisory. Validate package integrity post-installation. Reference: NIST SI-2 (Flaw Remediation), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management).
4. Step 4: Recovery, After patching, restart cert-manager pods and verify certificate issuance and renewal operations are functioning normally in all affected Kubernetes clusters. Confirm no pending certificate requests failed during the patching window and reissue any expired certificates. Monitor cert-manager process CPU and memory for 24-48 hours post-patch to confirm the infinite loop condition is resolved. Re-enable any SSH access restrictions relaxed during the maintenance window only after patch validation. Reference: NIST IR-4 (Incident Handling), NIST CP-10 (System Recovery and Reconstitution).
5. Step 5: Post-Incident, Review your Go dependency management process to ensure golang.org/x/crypto updates are tracked and applied within your patching SLA. Add cert-manager and golang.org/x/crypto/ssh to your continuous vulnerability monitoring scope. Evaluate whether SSH exposure of certificate management infrastructure requires additional network segmentation. Reference: NIST IR-8 (Incident Response Plan), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory).

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISO and cloud infrastructure leadership immediately if cert-manager CPU spike patterns are observed on production AKS clusters (indicating active exploitation of CVE-2026-39834), if certificate issuance failures cause TLS outages for customer-facing services, or if the patching window cannot be achieved within 24 hours due to change control constraints given the CVSS 9.1 rating and unauthenticated attack vector.
Recovery Notes	After patching azl3 cert-manager to the fixed version, prioritize validating the full certificate issuance pipeline by issuing a test certificate against each configured Issuer/ClusterIssuer in all affected Kubernetes namespaces — any cert-manager that was in an infinite loop state at time of patch may have corrupted in-flight certificate requests that require manual resubmission. Monitor cert-manager pod CPU utilization via 'kubectl top pod -n cert-manager' every 15 minutes for the first 4 hours post-restart and every hour thereafter for 48 hours total, treating any sustained CPU above 80% as a recurrence indicator requiring immediate re-isolation. Confirm Azure NSG rules restricting SSH to cert-manager nodes remain in place permanently and are codified in infrastructure-as-code (Terraform/Bicep) to prevent drift.
Forensic Artifacts	cert-manager pod crash logs from Kubernetes: 'kubectl logs -n cert-manager deployment/cert-manager --previous' — captures goroutine stack traces and any Go runtime output from the infinite loop condition within the golang.org/x/crypto/ssh channel write handler, which is the direct exploit mechanism for CVE-2026-39834. | Azure NSG Flow Logs for TCP/22 inbound to cert-manager nodes — identifies source IPs that sent oversized SSH_MSG_CHANNEL_DATA frames to trigger the infinite loop; filter for connections with abnormally high byte counts relative to session duration, which is the network-layer fingerprint of this specific DoS attack vector. | Linux kernel accounting records and systemd cgroup CPU stats: '/sys/fs/cgroup/system.slice/cert-manager.service/cpu.stat' or 'systemd-cgtop' output captured at time of incident — documents the infinite loop CPU consumption pattern specific to this vulnerability as a process-level forensic artifact. | RPM package database snapshot pre- and post-patch: 'rpm -qa --queryformat "%{NAME}-%{VERSION}-%{RELEASE}.%{ARCH} %{INSTALLTIME:date}\n"' filtered for azl3-cert-manager — establishes the version present during the exploitation window and the patch application timestamp for incident timeline reconstruction. | Go binary build information from the vulnerable cert-manager executable: 'go version -m /usr/bin/cert-manager' output preserved before patching — records the exact golang.org/x/crypto/ssh module version embedded in the affected binary, which is direct forensic evidence tying the installed software to the vulnerable code path in CVE-2026-39834.

Per-Action IR Details

Step 1: Containment — Identify all Azure Linux 3.0 hosts running azl3 cert-manager 1.12.15-6. Restrict SSH access to cert-manager nodes and any services directly exposing the golang.org/x/crypto/ssh library to untrusted networks. Apply network-layer controls (host firewall, NSG rules) to limit inbound SSH to known management IPs only. Reference: CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices).

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: Use Azure CLI to enumerate affected nodes: 'az vm list --query "[?storageProfile.imageReference.offer=='azurelinux']".name' combined with 'az vmss list' for AKS node pools. On each candidate node, run 'rpm -q azl3-cert-manager' via Azure Run Command to confirm version 1.12.15-6 without requiring SSH. Immediately update Azure NSG inbound rules via 'az network nsg rule update' to restrict TCP/22 to your bastion/jump host IP range only. If AKS, apply a Kubernetes NetworkPolicy to the cert-manager namespace

blocking all ingress except from the kube-system namespace and your defined management CIDR.

Evidence: Before applying NSG changes, capture current inbound SSH connection state: run 'ss -tnp sport = :22' on affected nodes to record active SSH sessions. Capture the cert-manager pod's current resource consumption with 'kubectl top pod -n cert-manager' and 'kubectl describe pod -n cert-manager' to establish a CPU/memory baseline. Export current Azure NSG effective rules via 'az network nic list-effective-nsg' for audit trail. Preserve any in-progress SSH session logs from /var/log/auth.log or journald ('journalctl -u ssh --since today') before locking down access.

Step 2: Detection — Query your asset inventory for Azure Linux 3.0 nodes with cert-manager installed. Run 'rpm -q azl3-cert-manager' or equivalent to confirm version 1.12.15-6 presence. Review process monitoring and system logs for cert-manager processes exhibiting high CPU utilization or hung states (indicative of infinite loop). Check SIEM for SSH connection anomalies: unusually large channel write payloads, repeated connection attempts to cert-manager endpoints, or process restarts. Reference: NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 8.2 (Collect Audit Logs), D3-SFA (System File Analysis), D3-LAM (Local Account Monitoring).

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

Compensating: Without a SIEM, run this inventory sweep via Azure Run Command across all AKS node pools: 'rpm -qa | grep cert-manager'. For process anomaly detection specific to the infinite loop condition, use: 'ps aux | grep cert-manager | awk "{print \$3, \$11}"' to identify processes pegged above 90% CPU. On the node, capture SSH channel write anomalies with 'tcpdump -i any -w /tmp/ssh_capture.pcap port 22 and tcp[32:4] > 32768' — the oversized channel write payload triggering CVE-2026-39834 will manifest as abnormally large SSH_MSG_CHANNEL_DATA frames. Use 'journalctl -u cert-manager -p err --since "24 hours ago"' to surface crash/restart events without a log aggregation platform.

Evidence: Collect the following before any remediation: (1) Full cert-manager pod logs via 'kubectl logs -n cert-manager deployment/cert-manager --previous' to capture crash output from any infinite-loop-induced OOMKill or process termination. (2) /var/log/auth.log or 'journalctl _COMM=sshd' entries showing SSH connection sources, timestamps, and any channel negotiation errors coinciding with cert-manager CPU spikes. (3) 'kubectl get events -n cert-manager --sort-by=.metadata.creationTimestamp' to establish timeline of pod restarts. (4) Go runtime panic output, if any, from systemd journal: 'journalctl -u cert-manager | grep -i "goroutine\|panic\|runtime"' — an infinite loop in golang.org/x/crypto/ssh may produce goroutine leak messages before process termination. (5) Network flow logs from Azure NSG Flow Logs showing source IPs sending repeated connections to TCP/22 on cert-manager nodes.

Step 3: Eradication — Apply the updated azl3 cert-manager package from the Microsoft Azure Linux 3.0 security repository as released under the May 2026 Patch Tuesday advisory (MSRC: CVE-2026-39834). For upstream Go projects using golang.org/x/crypto/ssh directly, update the dependency to the patched version identified in the upstream advisory. Validate package integrity post-installation. Reference: NIST SI-2 (Flaw Remediation), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST CM-3 (Configuration Change Control), NIST SA-10 (Developer Configuration Management), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Apply the patched azl3 cert-manager package via: 'tdnf update azl3-cert-manager -y' on each affected Azure Linux 3.0 node, pulling from the official Microsoft Azure Linux 3.0 security repository. Validate package integrity post-install with 'rpm -V azl3-cert-manager' — any output indicates file tampering. Confirm the embedded golang.org/x/crypto/ssh version is updated by extracting the binary's build info: 'go version -m /usr/bin/cert-manager | grep golang.org/x/crypto'. For any internally-built Go binaries consuming golang.org/x/crypto/ssh, update go.mod to the

patched version and rebuild, then verify with 'go mod verify'. Document the pre- and post-patch package hashes via 'rpm -q --queryformat "%{NAME} %{VERSION} %{SIGPGP:pgpsig}\n" azl3-cert-manager' for change control evidence.

Evidence: Before patching, capture the vulnerable binary's SHA-256 hash for change control and forensic comparison: 'sha256sum $(which cert-manager)'. Record the pre-patch package state: 'rpm -qi azl3-cert-manager > /tmp/certmgr_pre_patch.txt'. If exploitation is suspected, preserve a memory snapshot of any hung cert-manager process with 'gcore $(pgrep cert-manager)' before killing it — a goroutine dump may confirm the infinite loop stack trace within the golang.org/x/crypto/ssh channel write handler. Post-patch, re-run 'rpm -V azl3-cert-manager' and retain output as eradication evidence per NIST IR-5 (Incident Monitoring) documentation requirements.

Step 4: Recovery — After patching, restart cert-manager pods and verify certificate issuance and renewal operations are functioning normally in all affected Kubernetes clusters. Confirm no pending certificate requests failed during the patching window and reissue any expired certificates. Monitor cert-manager process CPU and memory for 24-48 hours post-patch to confirm the infinite loop condition is resolved. Re-enable any SSH access restrictions relaxed during the maintenance window only after patch validation. Reference: NIST IR-4 (Incident Handling), NIST CP-10 (System Recovery and Reconstitution).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST CP-10 (System Recovery and Reconstitution), NIST CP-9 (System Backup), CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: Restart cert-manager with: 'kubectl rollout restart deployment/cert-manager -n cert-manager' and watch rollout status with 'kubectl rollout status deployment/cert-manager -n cert-manager'. Verify certificate pipeline health by checking for failed CertificateRequests: 'kubectl get certificaterequest -A | grep -v Approved'. For any expired certificates caused by the DoS window, force renewal with: 'kubectl annotate certificate -n cert-manager.io/issueAs="$(date)"'. Set up a lightweight CPU watch without enterprise monitoring using a cron job: '*/5 * * * * ps aux | grep cert-manager >> /var/log/certmgr_cpu_watch.log' to capture any recurrence of the infinite loop CPU spike pattern over the 24-48 hour validation window.

Evidence: Document recovery evidence for the incident record: capture 'kubectl get certificate -A -o wide' output pre- and post-restart to confirm all certificates return to Ready=True state. Retain 'kubectl describe pod -n cert-manager' output after restart confirming no OOMKill events and normal memory/CPU allocation. Log the timestamp of NSG rule restoration and confirm no unauthorized SSH sources accessed cert-manager nodes during the maintenance window by reviewing Azure NSG Flow Logs for TCP/22 traffic outside the approved management CIDR. Preserve 48-hour CPU telemetry as evidence that the infinite loop condition (sustained cert-manager CPU at 100%) is resolved post-patch.

Step 5: Post-Incident — Review your Go dependency management process to ensure golang.org/x/crypto updates are tracked and applied within your patching SLA. Add cert-manager and golang.org/x/crypto/ssh to your continuous vulnerability monitoring scope. Evaluate whether SSH exposure of certificate management infrastructure requires additional network segmentation. Reference: NIST IR-8 (Incident Response Plan), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory).

NIST Phase: Post Incident

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

Controls: NIST IR-8 (Incident Response Plan), NIST IR-5 (Incident Monitoring), NIST SA-10 (Developer Configuration Management), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory)

Compensating: Integrate golang.org/x/crypto/ssh into your vulnerability tracking by adding a scheduled scan using 'govulncheck ./...' (free, official Go vulnerability scanner from Google) against all internal Go codebases and vendor directories on a weekly cron. For asset inventory coverage of cert-manager across AKS clusters, run 'kubectl get pods -A -o jsonpath="{range .items[*]}{.metadata.namespace} {.metadata.name} {.spec.containers[*].image}\n{end}" | grep cert-manager' as a weekly reconciliation script and pipe output to your CMDB or a version-controlled text file. Write a

Sigma rule targeting the infinite loop detection pattern (cert-manager process CPU > 90% for > 60 seconds) for integration into any future SIEM deployment. Create a JIRA/GitHub issue template for Go dependency CVEs requiring 'go mod tidy && go mod verify && govulncheck' as mandatory pre-closure steps.

Evidence: Retain the complete incident timeline — from first detection of cert-manager anomaly to confirmed patch validation — as a lessons-learned input per NIST 800-61r3 §4. Archive the pre-patch and post-patch 'rpm -qi' outputs, NSG rule change logs, certificate pipeline health checks, and CPU monitoring logs as the incident record. Conduct a dependency audit across all Azure Linux 3.0 workloads: 'find / -name go.sum 2>/dev/null | xargs grep golang.org/x/crypto' to identify any other internal services consuming the vulnerable library version, ensuring no residual exposure is undocumented.

Detection Guidance

Query your asset and software inventory for Azure Linux 3.0 nodes with azl3 cert-manager 1.12.15-6 installed (CIS 1.1, CIS 2.1). On identified hosts, check running processes for cert-manager instances consuming abnormal CPU; a hung infinite loop will typically peg a single CPU core at or near 100% without progressing. In your SIEM or log aggregation platform, search for: (1) SSH connection events to cert-manager service ports with unusually large payload sizes or connection durations; (2) process restart or crash events for cert-manager pods in Kubernetes audit logs (kubectl events or kube-apiserver audit logs); (3) certificate issuance failures or timeout errors in cert-manager controller logs, which would indicate the service became unresponsive. There are no published IOCs (IPs, hashes, domains) for this vulnerability at this time; exploitation would manifest as a service availability failure, not a data exfiltration pattern. NIST AU-6 and AU-12 apply for log collection and review cadence. D3-SFA (System File Analysis) and D3-LAM (Local Account Monitoring) are relevant defensive techniques for baseline process behavior monitoring.

Framework Mappings

MITRE-ATTACK

- **T1499.004** — Application or System Exploitation

CIS-V8

- **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.8.24** — Use of cryptography
- **A.5.23** — Information security for use of cloud services

NIST-800-53R5

- **SC-13** — Cryptographic Protection

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1499.004	Application or System Exploitation	Impact

Sources

Source	URL	Tier
MSRC Update Guide	https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-39834	T1
(consolidated)	https://api.msrc.microsoft.com/cvrf/v3.0/cvrf/2026-May	T1
CVE-2026-39834 | Mondoo Vulnerability Intelligence	https://mondoo.com/vulnerability-intelligence/vulnerability/CVE-202...	T3
CVE-2026-39834 - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-39834	T1
CVE-2026-39834 Common Vulnerabilities and Exposures | SUSE	https://www.suse.com/security/cve/CVE-2026-39834.html	T3

Generated 2026-05-27 06:39 UTC by TJS Security Command Center