INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-27 06:40 UTC

Auth Bypass via Unenforced @revoked Status in golang.org/x/crypto/ssh/knownhosts (CVE-2026-42508)

CVE VULNERABILITY | CRITICAL | CVSS 9.1

SCC Item ID	SCC-CVE-2026-0229
Type	CVE Vulnerability
CVE ID	CVE-2026-42508
Severity	CRITICAL
CVSS Base Score	9.1
EPSS Score	0.0003 (9th percentile)
Affected Products	Microsoft Azure Linux 3.0, azl3 libcontainers-common 20240213-3; underlying vulnerability in golang.org/x/crypto/ssh/knownhosts
Published	2026-05-27T01:40:27
Discovery Source	Msrc Patch Tuesday

Executive Summary

A critical authentication bypass vulnerability (CVSS 9.1) in the Go SSH known_hosts library allows attackers to present revoked host keys and pass authentication checks undetected. Microsoft disclosed this as CVE-2026-42508 during Patch Tuesday May 2026; the affected package ships in Azure Linux 3.0 via the azl3 libcontainers-common bundle. Organizations relying on SSH host verification in containerized workloads on Azure Linux 3.0 face exposure to man-in-the-middle and impersonation attacks until the patch is applied.

Technical Analysis

CVE-2026-42508 affects golang.org/x/crypto/ssh/knownhosts, the Go standard library package used to parse and enforce SSH known_hosts files. The @revoked marker, intended to flag host keys that must be rejected, is not enforced by the library. A host presenting a key marked @revoked will successfully complete SSH host authentication, defeating the revocation mechanism entirely. CWE-285 (Improper Authorization) and CWE-862 (Missing Authorization) describe the root cause. CVSS base score is 9.1; EPSS is 0.03% (8.83rd percentile) as of disclosure, indicating low observed exploitation activity but high theoretical impact. The vulnerability surfaces in Microsoft Azure Linux 3.0 through the azl3 libcontainers-common package, which bundles the vulnerable knownhosts implementation. MITRE ATT&CK techniques T1550 (Use Alternate Authentication Material) and

T1557 (Adversary-in-the-Middle) describe the relevant attack paths. No CISA KEV listing as of configuration date. Patch available via Microsoft MSRC advisory for CVE-2026-42508; upstream Go fix expected in golang.org/x/crypto. No public exploit code confirmed.

Action Checklist

1. Step 1: Containment. Identify all Azure Linux 3.0 systems running azl3 libcontainers-common versions prior to the patched release. Restrict SSH-dependent container workloads on unpatched hosts from untrusted networks until the fix is applied. Reference MSRC advisory at https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-42508 for affected package version thresholds. Apply NIST AC-17 (Remote Access) controls to limit exposure of SSH endpoints during the remediation window.
2. Step 2: Detection. Query package managers on Azure Linux 3.0 hosts: run 'tdnf list installed libcontainers-common' and compare against the patched version listed in the MSRC advisory. Review SSH connection logs (auth.log, journald SSH entries) for successful authentications from hosts whose keys appear in known_hosts files with the @revoked marker. Flag any SSH session where the connecting host key matches a @revoked entry; these should have been rejected. Align log collection with NIST AU-2 (Event Logging) and CIS 8.2 (Collect Audit Logs) to ensure SSH authentication events are captured.
3. Step 3: Eradication. Apply the patched azl3 libcontainers-common package via 'tdnf update libcontainers-common' on all Azure Linux 3.0 hosts, per the MSRC advisory for CVE-2026-42508. After patching, audit all SSH known_hosts files across the environment and confirm that @revoked entries are present for any keys that have been revoked. Rotate any SSH host keys that may have been exposed or used during the vulnerability window. Reference NIST SI-2 (Flaw Remediation) and CIS 7.3 (Perform Automated Operating System Patch Management).
4. Step 4: Recovery. After patching, test SSH connections from systems that rely on knownhosts-based verification to confirm that hosts presenting @revoked keys are now rejected. Monitor SSH authentication logs for anomalous patterns for at least 30 days post-remediation. Validate that container orchestration pipelines dependent on SSH host verification resume normal operation. Apply NIST AU-6 (Audit Record Review, Analysis, and Reporting) cadence to SSH log review during this window.
5. Step 5: Post-Incident. This vulnerability exposes a gap in reliance on SSH revocation as a trust control without validating enforcement. Implement periodic testing of known_hosts @revoked enforcement as part of security regression testing in CI/CD pipelines. Review all Go-based internal tooling that imports golang.org/x/crypto/ssh/knownhosts for the same defect pattern. Add dependency version pinning and automated vulnerability scanning for Go modules per CIS 2.2 (Ensure Authorized Software is Currently Supported). Apply NIST AC-2 (Account Management) and SC-2 (Access Agreements) by rotating SSH host keys on a defined schedule and enforcing key revocation through a validated, tested mechanism.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to senior IR leadership and legal/compliance if SSH auth log analysis from Step 2 identifies any session where a @revoked host key was accepted — confirming active exploitation — particularly if the authenticating container workload handled PII, PHI, or secrets (triggering breach notification obligations under applicable regulation), or if the organization lacks visibility into SSH auth logs for the relevant window.
Recovery Notes	After patching azl3 libcontainers-common and rotating SSH host keys, functionally verify that the knownhosts @revoked enforcement now correctly rejects revoked keys using the scripted test described in Step 4 before re-exposing container workloads to untrusted networks. Monitor sshd authentication journals daily for 30 days specifically for any authentication attempts using the old (rotated) host key fingerprints, which would indicate a threat actor attempting to reconnect using a key cached from a prior session during the vulnerability window. Confirm all Go-based container images that embed golang.org/x/crypto/ssh/knownhosts have been rebuilt against the patched module version, not merely that the host OS package was updated.
Forensic Artifacts	Per-host SSH daemon journal exports ('journalctl -u sshd -o json') covering the full vulnerability exposure window — specifically entries where 'Accepted' authentication events can be cross-referenced against @revoked fingerprints in the known_hosts files present at time of authentication, which is the direct forensic signature of CVE-2026-42508 exploitation. | Forensic copies (with sha256 checksums) of all '/etc/ssh/ssh_known_hosts' and '~/.ssh/known_hosts' files from affected Azure Linux 3.0 hosts at time of detection — these files contain the @revoked entries that should have blocked authentication and are required to determine whether any accepted session fingerprint matched a revoked key. | Azure NSG flow logs or host-level tcpdump captures for TCP/22 to affected hosts during the exposure window — these provide the external IP addresses of SSH clients that connected, enabling correlation with threat intelligence feeds to identify potential MITM or impersonation actors who may have used CVE-2026-42508 to present a revoked host key. | Output of 'rpm -qa --queryformat "%{NAME} %{VERSION} %{INSTALLTIME:date}\n" | grep libcontainers-common' from each affected host, establishing the exact vulnerable package version and installation timestamp — this determines the precise start of each host's exposure window for scoping any breach notification timeline. | Go module dependency graphs ('go mod graph' output or 'go.sum' files) from all internal container image build repositories, identifying every binary artifact that statically linked the vulnerable golang.org/x/crypto/ssh/knownhosts version and was deployed into production during the exposure window — these artifacts define the full blast radius beyond the azl3 OS package itself.

Per-Action IR Details

Step 1: Containment — Identify all Azure Linux 3.0 systems running azl3 libcontainers-common versions prior to the patched release. Isolate or restrict SSH-dependent container workloads on unpatched hosts from untrusted networks until the fix is applied. Reference MSRC advisory at https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-42508 for affected package version thresholds. Apply NIST AC-17 (Remote Access) controls to limit exposure of SSH endpoints during the remediation window.

NIST Phase: Containment

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

Controls: NIST AC-17 (Remote Access), NIST AC-4 (Information Flow Enforcement), NIST IR-4 (Incident Handling), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory)

Compensating: Run 'tdnf list installed libcontainers-common' via a bash loop across all Azure Linux 3.0 hosts using SSH with a known-good key (bypassing knownhosts verification temporarily via 'StrictHostKeyChecking=no' only on a trusted internal management VLAN): for host in $(cat hosts.txt); do ssh azureuser@$host 'tdnf list installed libcontainers-common'; done. Block inbound SSH (TCP/22) to unpatched container hosts at the host firewall using 'firewall-cmd --add-rich-rule="rule family=ipv4 source address=0.0.0.0/0 port port=22 protocol=tcp reject"' except from your designated jump host IP. Use osquery 'SELECT * FROM rpm_packages WHERE name="libcontainers-common";' for bulk inventory if osquery is deployed.

Evidence: Before isolating, snapshot the current SSH known_hosts files from all affected hosts — specifically '/etc/ssh/ssh_known_hosts' and per-user '~/.ssh/known_hosts' — and preserve them as forensic copies with checksums (sha256sum). Capture 'journalctl -u sshd --since "72 hours ago" > sshd_pre_containment.log' on each host to preserve SSH daemon authentication records before any network changes could cause log rotation or loss. Document all container workloads using SSH host verification by querying 'crictl ps' and correlating with container image manifests that reference golang.org/x/crypto/ssh/knownhosts.

Step 2: Detection — Query package managers on Azure Linux 3.0 hosts: run 'tdnf list installed libcontainers-common' and compare against the patched version listed in the MSRC advisory. Review SSH connection logs (auth.log, journald SSH entries) for successful authentications from hosts whose keys appear in known_hosts files with the @revoked marker. Flag any SSH session where the connecting host key matches a @revoked entry — these should have been rejected. Align log collection with NIST AU-2 (Event Logging) and CIS 8.2 (Collect Audit Logs) to ensure SSH authentication events are captured.

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: Extract all @revoked entries from known_hosts files: 'grep -r "@revoked" /etc/ssh/ssh_known_hosts ~/.ssh/known_hosts 2>/dev/null | awk "{print $2}" > revoked_keys.txt'. Then parse SSH auth logs for accepted sessions and cross-reference connecting host fingerprints: 'journalctl -u sshd | grep "Accepted" | grep -oP "from \K[^]+" > accepted_ips.txt'. For each accepted IP, retrieve its current host key fingerprint using 'ssh-keyscan -t rsa,ecdsa,ed25519 2>/dev/null | ssh-keygen -lf -' and compare against revoked_keys.txt. Write a Sigma rule targeting sshd journal entries where authentication succeeded from a host fingerprint matching a @revoked known_hosts entry. Use 'awk' to diff accepted fingerprints against revoked list in a scheduled cron job running every 15 minutes during the incident window.

Evidence: Preserve 'journalctl -u sshd -o json > sshd_json.log' (JSON format retains structured fields including remote host, accepted key fingerprint, and timestamp) from each Azure Linux 3.0 host. Extract and hash all known_hosts files system-wide ('find / -name known_hosts 2>/dev/null -exec sha256sum {} \;') to establish a forensic baseline of which @revoked entries existed at time of detection. Capture network flow data (if available via Azure NSG flow logs or tcpdump) for TCP/22 connections to affected hosts in the 90 days prior to detection, to identify any sessions that may have exploited the bypass. Pull container runtime logs ('journalctl -u containerd' or 'journalctl -u docker') for any SSH-initiated container operations that correlate in time with suspicious authentication events.

Step 3: Eradication — Apply the patched azl3 libcontainers-common package via 'tdnf update libcontainers-common' on all Azure Linux 3.0 hosts, per the MSRC advisory for CVE-2026-42508. After patching, audit all SSH known_hosts files across the environment and confirm that @revoked entries are present for any keys that have been revoked. Rotate any SSH host keys that may have been exposed or used during the vulnerability window. Reference NIST SI-2 (Flaw Remediation) and CIS 7.3 (Perform Automated Operating System Patch Management).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST CM-7 (Least Functionality), NIST IA-3 (Device Identification and Authentication), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 5.2 (Use Unique Passwords)

Compensating: Execute patching in a documented sequence: (1) 'tdnf update libcontainers-common' on each host; (2) verify with 'tdnf list installed libcontainers-common' that the version matches or exceeds the fixed version from the MSRC advisory; (3) rotate SSH host keys by stopping sshd, deleting '/etc/ssh/ssh_host_*', running 'ssh-keygen -A' to regenerate, then restarting sshd: 'systemctl stop sshd && rm /etc/ssh/ssh_host_* && ssh-keygen -A && systemctl start sshd'; (4) redistribute new host fingerprints to all known_hosts files via 'ssh-keyscan' from a trusted host and replace old entries. For Go-based internal tooling, run 'grep -r "golang.org/x/crypto/ssh/knownhosts" /path/to/repos --include="go.sum" --include="go.mod"' to identify all affected binaries requiring recompilation against the patched module version.

Evidence: Before patching, preserve the pre-patch binary of the vulnerable libcontainers-common package: 'rpm -ql libcontainers-common | xargs sha256sum > pre_patch_hashes.txt' and copy the package RPM itself to an evidence store. Capture 'tdnf history' to document the patch transaction with timestamp and actor. After host key rotation, archive the old public host keys (already collected in Step 1 known_hosts snapshots) as forensic evidence of what keys were trusted during the vulnerability window — these are needed if post-incident analysis reveals a MITM session occurred using a now-revoked key that was previously @revoked but accepted due to the bypass.

Step 4: Recovery — After patching, test SSH connections from systems that rely on knownhosts-based verification to confirm that hosts presenting @revoked keys are now rejected. Monitor SSH authentication logs for anomalous patterns for at least 30 days post-remediation. Validate that container orchestration pipelines dependent on SSH host verification resume normal operation. Apply NIST AU-6 (Audit Record Review, Analysis, and Reporting) cadence to SSH log review during this window.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-3 (Content of Audit Records), NIST CP-4 (Contingency Plan Testing), NIST SI-6 (Security and Privacy Function Verification), CIS 8.2 (Collect Audit Logs)

Compensating: Functionally verify the patch corrects the bypass: create a test known_hosts file with a @revoked entry for a test host's actual key, then attempt an SSH connection to that host using the Go-based tooling or 'ssh -o UserKnownHostsFile=test_known_hosts testuser@testhost' — the connection MUST be rejected with 'Host key verification failed' to confirm the fix. Script this as a daily cron verification for 30 days: 'ssh -o StrictHostKeyChecking=yes -o UserKnownHostsFile=/path/to/test_known_hosts -o BatchMode=yes testuser@testhost exit 2>&1 | grep -q "Host key verification failed" && echo PASS || echo FAIL >> /var/log/ssh_revocation_test.log'. Set up 'journalctl -f -u sshd | grep -E "(Accepted|Failed|Invalid)"' piped to a log aggregator or local file with daily review cadence.

Evidence: Document the functional verification test results (pass/fail with timestamps) as formal evidence of remediation effectiveness per NIST 800-61r3 §3.5 recovery validation requirements. Capture 'rpm -q --changelog libcontainers-common' post-patch to record the exact package version and changelog confirming CVE-2026-42508 fix. Retain SSH auth logs (journald exports) for the 30-day monitoring window with integrity protection ('journalctl --verify' output archived weekly) to support any downstream forensic analysis if a breach is later confirmed.

Step 5: Post-Incident — This vulnerability exposes a gap in reliance on SSH revocation as a trust control without validating enforcement. Implement periodic testing of known_hosts @revoked enforcement as part of security regression testing in CI/CD pipelines. Review all Go-based internal tooling that imports golang.org/x/crypto/ssh/knownhosts for the same defect pattern. Add dependency version pinning and automated vulnerability scanning for Go modules per CIS 2.2 (Ensure Authorized Software is Currently Supported). Apply D3-CH (Credential Hardening) by rotating SSH host keys on a defined schedule and enforcing key revocation through a validated, tested mechanism.

NIST Phase: Post Incident

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

Controls: NIST SI-2 (Flaw Remediation), NIST CA-7 (Continuous Monitoring), NIST SA-11 (Developer Testing and Evaluation), NIST RA-5 (Vulnerability Monitoring and Scanning), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Integrate Go module vulnerability scanning into CI/CD using 'govulncheck ./...' (free, official Go vulnerability checker from Google) against all internal repositories that import 'golang.org/x/crypto/ssh/knownhosts' — this would have flagged CVE-2026-42508 before deployment. Add a 'go.sum' integrity check step: 'go mod verify' in pipeline gates. Write a regression test specifically for @revoked enforcement: include it in the repo test suite as a Go test that instantiates a knownhosts.New() parser with a @revoked entry and asserts that HostKeyCallback returns an error for that key — this test must pass on every build. Use YARA to scan container images for binaries linked against the vulnerable golang.org/x/crypto version: write a rule matching the vulnerable knownhosts.go function signature bytes and run it against image layers via 'yara rule.yar /path/to/extracted/layer'.

Evidence: Produce a lessons-learned document specifically addressing: (1) how long @revoked entries existed in known_hosts files without enforcement being tested, (2) which container workloads were running Go tooling importing the vulnerable knownhosts package during the exposure window, and (3) whether any SSH authentication events during the window involved hosts with @revoked keys (from the log analysis in Step 2). Archive the pre-patch known_hosts snapshots, the sshd journal exports, and the vulnerability scan output from govulncheck as the evidentiary record for this incident. If regulatory obligations apply (e.g., FedRAMP, PCI-DSS for Azure-hosted workloads), retain this package as the incident closure documentation.

Detection Guidance

On Azure Linux 3.0 hosts, run 'tdnf list installed libcontainers-common' to identify unpatched versions. To detect exploitation attempts, parse SSH daemon logs (typically /var/log/auth.log or journald with 'sshd' unit filter) for successful authentication events where the source host key matches a @revoked entry in any known_hosts file on the system. A successful auth from a @revoked key is the primary behavioral indicator; these events should not occur on patched systems. Cross-reference known_hosts files at ~/.ssh/known_hosts and /etc/ssh/ssh_known_hosts for @revoked entries and build a watchlist of those key fingerprints. Alert on SSH sessions authenticated by any key fingerprint on that watchlist. Align detection logging with NIST AU-2 (Event Logging), AU-3 (Content of Audit Records), and CIS 8.2 (Collect Audit Logs) to ensure SSH auth events include timestamp, source IP, and key fingerprint. No public IOC hashes or IPs associated with active exploitation as of disclosure.

Framework Mappings

MITRE-ATTACK

- **T1550** — Use Alternate Authentication Material
- **T1557** — Adversary-in-the-Middle

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control

NIST-800-53R5

- **AC-3** — Access Enforcement
- **IA-2** — Identification and Authentication (Organizational Users)

CIS-V8

- **6.1** — Establish an Access Granting Process
- **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
- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1550	Use Alternate Authentication Material	Defense-Evasion
T1557	Adversary-in-the-Middle	Credential-Access

Sources

Source	URL	Tier
MSRC Update Guide	https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-42508	T1
(consolidated)	https://api.msrc.microsoft.com/cvrf/v3.0/cvrf/2026-May	T1
CVE-2026-42508 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-42508	T1
Linux Distros Unpatched Vulnerability : CVE-2026-42508 | Tenable®	https://www.tenable.com/plugins/nessus/316602	T3
Amazon Linux Security Center - CVE List	https://explore.alas.aws.amazon.com/	T3

Generated 2026-05-27 06:40 UTC by TJS Security Command Center