INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-27 06:38 UTC

CVE-2026-46595: Critical golang.org/x/crypto/ssh VerifiedPublicKeyCallback Permission Bypass in Microsoft Azure Linux Package

CVE VULNERABILITY | **CRITICAL** | CVSS 10.0

SCC Item ID	SCC-CVE-2026-0222
Type	CVE Vulnerability
CVE ID	CVE-2026-46595
Severity	CRITICAL
CVSS Base Score	10.0
EPSS Score	0.0004 (12th percentile)
Affected Products	Microsoft azl3 docker-buildx 0.14.0-11 on Azure Linux 3.0 (golang.org/x/crypto/ssh)
Published	2026-05-27T01:40:27
Discovery Source	Msrc Patch Tuesday

Executive Summary

A critical authentication bypass vulnerability (CVSS 10.0) in the Go cryptography library's SSH implementation allows an attacker to bypass public key authentication controls entirely when the VerifiedPublicKeyCallback mechanism is invoked. The affected component is Microsoft's azl3 docker-buildx 0.14.0-11 package on Azure Linux 3.0, used in container build infrastructure. Organizations running this package in CI/CD or container build pipelines are exposed to unauthorized access to build systems, with potential for supply chain compromise.

Technical Analysis

CVE-2026-46595 affects golang.org/x/crypto/ssh as shipped in Microsoft's azl3 docker-buildx 0.14.0-11 package on Azure Linux 3.0. The flaw resides in the VerifiedPublicKeyCallback mechanism: when this callback is invoked during SSH handshake processing, permissions enforcement is skipped entirely, allowing a client presenting any public key to gain access without valid authorization. Associated CWEs: CWE-863 (Incorrect Authorization) and CWE-284 (Improper Access Control). Relevant MITRE ATT&CK techniques: T1650 (Acquire Access), T1078 (Valid Accounts), T1556 (Modify Authentication Process). CVSS base score is 10.0; EPSS score is 0.0004 (12th percentile), indicating low observed exploitation activity as of disclosure. Not currently listed on CISA KEV. Microsoft released a patch in May 2026 (Patch Tuesday). Patch status: check MSRC

advisory at https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-46595 and NVD at https://nvd.nist.gov/vuln/detail/CVE-2026-46595 for patch release and apply immediately when available.

Action Checklist

1. Step 1: Containment. Immediately identify all Azure Linux 3.0 hosts running docker-buildx 0.14.0-11 (package name: azl3 docker-buildx). Restrict SSH access to those build hosts from untrusted networks using host-based firewall rules (CIS 4.4) until a patched package is available.
2. Step 2: Detection. Query package inventory for 'docker-buildx 0.14.0-11' on Azure Linux 3.0 hosts. Review SSH authentication logs (typically /var/log/auth.log or journalctl -u sshd) for unexpected successful logins, especially those not matching known key fingerprints or originating from unexpected source IPs. Enable audit logging per NIST AU-2 and AU-12 if not already active. Correlate against MITRE T1078 (Valid Accounts) and T1650 (Acquire Access) behavioral patterns: look for new or unknown accounts accessing build hosts post-authentication.
3. Step 3: Eradication. Apply the updated azl3 docker-buildx package from Microsoft's Azure Linux 3.0 repository when released. Monitor the MSRC advisory (https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-46595) for patch availability and apply per CIS 7.3 (Perform Automated Operating System Patch Management) and CIS 7.4 (Perform Automated Application Patch Management). If your build tooling supports it, switch to certificate-based or alternative SSH authentication methods that do not rely on VerifiedPublicKeyCallback.
4. Step 4: Recovery. After patching, rotate all SSH keys associated with affected build hosts (NIST IA controls; D3-CRO: Credential Rotation). Verify the updated package version is confirmed installed. Re-enable full SSH access only after confirming the patched version is in place. Monitor SSH authentication logs for 72 hours post-remediation for anomalous access attempts. Review build artifacts produced during the exposure window for signs of tampering (D3-SFA: System File Analysis).
5. Step 5: Post-Incident. Audit the CI/CD pipeline for least-privilege access controls (NIST AC-6) to limit blast radius if a build host is compromised. Implement account monitoring (D3-LAM: Local Account Monitoring) on build infrastructure. Review credential hardening posture (D3-CH) for SSH key management. Establish automated package inventory scanning (CIS 2.1) to detect future vulnerable package deployments before they reach production build environments. Evaluate whether SSH public key authentication configurations across other Go-based services use VerifiedPublicKeyCallback and assess exposure breadth.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO and legal/compliance if SSH authentication logs show any `Accepted publickey` events with unrecognized key fingerprints on affected docker-buildx 0.14.0-11 hosts during the exposure window, if container images built on those hosts were pushed to production registries or delivered to customers (triggering potential software supply chain breach notification), or if the incident response team lacks capability to audit build artifact integrity across the exposure window.

Recovery Notes	After applying the patched azl3 docker-buildx package, verify remediation by confirming `tdnf info docker-buildx` reflects the fixed version and running `rpm -V docker-buildx` to validate package file integrity. All container images built on affected hosts between the docker-buildx 0.14.0-11 install date and the patch date must be treated as potentially compromised supply chain artifacts and re-built from verified clean hosts before redeployment to production. Maintain enhanced SSH authentication log monitoring (`journalctl -u sshd -f`) for a minimum of 72 hours post-patch to detect any attacker persistence mechanisms (authorized_keys backdoors, SSH daemon modifications) that may have been installed during the exploitation window.
Forensic Artifacts	/var/log/auth.log or `journalctl -u sshd -o json` covering the full exposure window: specifically `Accepted publickey` entries where the authenticating key fingerprint is absent from all `authorized_keys` files on the host — this is the direct on-disk forensic signature of a VerifiedPublicKeyCallback bypass exploiting CVE-2026-46595. | /var/lib/buildkit/ directory and build cache manifests: preserves the complete record of container build operations performed by the compromised docker-buildx 0.14.0-11 process, including any injected build steps or tampered layer digests that would indicate supply chain compromise post-authentication bypass. | Pre-patch SHA-256 hash of the docker-buildx binary (`sha256sum $(which docker-buildx)`) and `rpm -qi docker-buildx` install timestamp: establishes the exact exposure window and provides a cryptographic baseline for confirming the vulnerable golang.org/x/crypto/ssh version was present in the production binary. | Output of `ss -tnp | grep :22` and `/proc//net/tcp` snapshots taken at time of containment: reveals any active or recently established SSH port forwards or reverse tunnels (MITRE T1572) that an attacker may have created through the authentication bypass before the host was isolated. | Container registry push logs (Azure Container Registry activity logs or local Docker daemon logs at `/var/log/docker.log` or `journalctl -u docker`) for the exposure window: identifies which potentially compromised build artifacts were promoted to downstream registries or production pipelines, scoping the supply chain impact radius of the CVE-2026-46595 exploitation.

Per-Action IR Details

Step 1: Containment — Immediately identify all Azure Linux 3.0 hosts running docker-buildx 0.14.0-11 (package name: azl3 docker-buildx). Isolate or restrict SSH access to those build hosts from untrusted networks until a patched package is available. Enforce host-based firewall rules (CIS 4.4) to limit SSH exposure to authorized management networks only.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST AC-4 (Information Flow Enforcement), 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 | grep docker-buildx` on all Azure Linux 3.0 hosts via a bash loop over your SSH inventory (e.g., `for host in $(cat hosts.txt); do ssh $host 'tdnf list installed | grep docker-buildx'; done`) to enumerate affected nodes. Immediately apply an iptables or nftables rule restricting SSH (port 22) to your management CIDR only: `iptables -I INPUT -p tcp --dport 22 ! -s -j DROP`. For teams using osquery, deploy `SELECT * FROM deb_packages WHERE name LIKE '%docker-buildx%';` via osquery fleet to enumerate at scale without an enterprise SIEM.

Evidence: Before isolating, capture the current SSH daemon configuration (`sshd_config`) to document whether `VerifiedPublicKeyCallback`-based auth paths are explicitly configured in the build host's SSH server or invoked by docker-buildx daemon processes. Preserve the current active SSH session table (`ss -tnp | grep :22`) and `who -a` output to identify any sessions already established through the bypass. Document iptables/nftables state pre-change so post-incident analysis can confirm what was previously reachable.

Step 2: Detection — Query package inventory for 'docker-buildx 0.14.0-11' on Azure Linux 3.0 hosts. Review SSH authentication logs (typically /var/log/auth.log or journalctl -u sshd) for unexpected successful logins, especially those not matching known key fingerprints or originating from unexpected source IPs. Enable audit logging per NIST AU-2 and AU-12 if not already active. Correlate against MITRE T1078 (Valid Accounts) and T1650 (Acquire Access) behavioral patterns: look for new or unknown accounts accessing build hosts post-authentication.

NIST Phase: Detection Analysis

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

Controls: NIST AU-2 (Event Logging), NIST AU-12 (Audit Record Generation), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: On each Azure Linux 3.0 build host, run: `journalctl -u sshd --since '30 days ago' | grep -E 'Accepted publickey|Invalid user|Failed|session opened'` to surface authentication events. The CVE-2026-46595 bypass would cause `Accepted publickey` entries for keys that are NOT in `~/.ssh/authorized_keys` on the target account — cross-reference with: `grep 'Accepted publickey' /var/log/auth.log | awk '{print $9, $11}' | sort | uniq` and diff against your known-good key fingerprint inventory. Deploy the Sigma rule targeting T1078 (sshd anomalous auth) using `sigma convert` to a grep-compatible format for hosts without SIEM. Enable Linux auditd rules: `auditctl -w /var/log/auth.log -p war -k ssh_auth_monitor` and `auditctl -a always,exit -F arch=b64 -S execve -F uid=0 -k root_exec` to capture post-auth lateral movement from the build user context.

Evidence: Collect `/var/log/auth.log` (or `journalctl -u sshd -o json` for structured output) covering the full exposure window since docker-buildx 0.14.0-11 was installed — check install date via `rpm -qi docker-buildx` or `tdnf history`. Specifically hunt for `Accepted publickey` log lines where the key fingerprint does not appear in any authorized_keys file on the system, which is the direct forensic signature of a VerifiedPublicKeyCallback bypass. Also collect `/proc//net/tcp` and `netstat -anp | grep sshd` snapshots to identify any persistent reverse tunnels or port forwards established via an exploited session, consistent with MITRE T1572 (Protocol Tunneling).

Step 3: Eradication — Apply the updated azl3 docker-buildx package from Microsoft's Azure Linux 3.0 repository when released. Monitor the MSRC advisory (https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-46595) for patch availability and apply per CIS 7.3 (Perform Automated Operating System Patch Management) and CIS 7.4 (Perform Automated Application Patch Management). Until patched, consider replacing VerifiedPublicKeyCallback-based SSH authentication paths with alternative authentication flows if the build tooling supports it.

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), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Poll the Azure Linux 3.0 package mirror for patch availability using: `tdnf check-update docker-buildx 2>&1 | grep docker-buildx` — run this on a cron schedule (every 4 hours) until the fix lands. Once available, apply with: `tdnf update docker-buildx -y && tdnf verify docker-buildx` and capture the output as remediation evidence. For the interim compensating control replacing VerifiedPublicKeyCallback: audit your docker-buildx daemon configuration files (typically under `/etc/docker/` or `~/.docker/buildx/`) for any SSH transport configuration referencing custom callback handlers, and if present, revert to standard `AuthorizedKeysFile`-backed public key auth by restarting the SSH service with a stripped-down `sshd_config` that disables `PubkeyAuthentication` entirely until the patch is applied — forcing certificate-based or password auth as a temporary stopgap in isolated management networks only.

Evidence: Before applying the patch, snapshot the installed package state with `rpm -qa --last | grep docker-buildx` and retain the pre-patch binary hash: `sha256sum $(which docker-buildx)`. This establishes a forensic baseline to prove pre-patch vs. post-patch state and supports any regulatory reporting requirement. Also preserve a copy of `/usr/lib/golang/src/golang.org/x/crypto/ssh/` (if accessible) or the vendored crypto library path within the docker-buildx binary to document the vulnerable golang.org/x/crypto version present, confirming scope of the CVE-2026-46595 affected code path.

Step 4: Recovery — After patching, rotate all SSH keys associated with affected build hosts (NIST IA controls; D3-CRO: Credential Rotation). Verify the updated package version is confirmed installed. Re-enable full SSH access only after confirming the patched version is in place. Monitor SSH authentication logs for 72 hours post-remediation for anomalous access attempts. Review build artifacts produced during the exposure window for signs of tampering (D3-SFA: System File Analysis).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IA-5 (Authenticator Management), NIST IA-3 (Device Identification and Authentication), NIST CM-2 (Baseline Configuration), CIS 5.2 (Use Unique Passwords), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Automate SSH key rotation using a bash script that iterates over `/home/*/.ssh/authorized_keys` and `/root/.ssh/authorized_keys`, revokes all existing entries, generates new ED25519 keypairs via `ssh-keygen -t ed25519 -f /tmp/new_build_key -N "`, and re-deploys only to verified administrator accounts. For build artifact integrity verification during the exposure window, compute SHA-256 hashes of all container images produced by the affected docker-buildx hosts using `docker images --format '{{.ID}}' | xargs -I{} docker inspect {} | jq '.[].RootFS'` and compare against your CI/CD pipeline's expected digest values stored in your registry or pipeline logs. Use `rpm -V docker-buildx` post-patch to verify no package files were tampered with.

Evidence: Collect all container image digests and build logs generated by the affected docker-buildx 0.14.0-11 hosts during the entire exposure window (from package install date to patch date) from your container registry (e.g., Azure Container Registry audit logs, or local `/var/lib/docker/` layer hashes). A successful CVE-2026-46595 exploitation of the build host could result in injected malicious layers or modified build steps — preserve `docker buildx inspect` output and build cache manifests (`/var/lib/buildkit/`) as forensic evidence of what was built under potentially compromised conditions.

Step 5: Post-Incident — Audit the CI/CD pipeline for least-privilege access controls (NIST AC-6) to limit blast radius if a build host is compromised. Implement account monitoring (D3-LAM: Local Account Monitoring) on build infrastructure. Review credential hardening posture (D3-CH) for SSH key management. Establish automated package inventory scanning (CIS 2.1) to detect future vulnerable package deployments before they reach production build environments. Evaluate whether SSH public key authentication configurations across other Go-based services use VerifiedPublicKeyCallback and assess exposure breadth.

NIST Phase: Post Incident

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

Controls: NIST AC-6 (Least Privilege), NIST AC-2 (Account Management), NIST RA-3 (Risk Assessment), NIST SI-2 (Flaw Remediation), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Deploy osquery on all build infrastructure with a scheduled query to detect future vulnerable package states: `SELECT name, version FROM deb_packages WHERE name LIKE '%docker-buildx%' OR name LIKE '%golang%';` — pipe results to a log file reviewed daily. To enumerate other Go services using VerifiedPublicKeyCallback across your environment, run: `grep -r 'VerifiedPublicKeyCallback' /usr/local/go/ /opt/ /home/ /srv/ 2>/dev/null` on each host to identify any custom Go SSH implementations that may share the same vulnerable pattern from golang.org/x/crypto/ssh. For CI/CD least-privilege hardening, audit your pipeline service account permissions using: `getent group docker sudo` and `id ` to confirm build users have no unnecessary sudo or group escalation paths that would amplify a future SSH authentication bypass.

Evidence: Produce a post-incident lessons-learned artifact documenting: (1) the full list of docker-buildx 0.14.0-11 hosts confirmed affected, (2) the exposure window duration derived from `rpm -qi docker-buildx` install timestamps, (3) all SSH authentication log anomalies found or ruled out during the investigation, and (4) inventory of all container artifacts built during the exposure window and their integrity verification status. This record supports both internal risk acceptance decisions and potential regulatory disclosure if any build artifacts were distributed to production systems or customers.

Detection Guidance

Query package managers on all Azure Linux 3.0 hosts: 'tdnf list installed | grep docker-buildx' or equivalent. Flag any result returning version 0.14.0-11. Review SSH daemon logs (/var/log/auth.log, /var/log/secure, or 'journalctl -u sshd') for successful authentications that do not correspond to pre-approved key fingerprints. Look for authentication events where the accepted key fingerprint does not match entries in authorized_keys files. Monitor for new or previously unseen source IPs authenticating successfully to build hosts (MITRE T1650, T1078). Enable NIST AU-2 event logging for SSH authentication success and failure events if not already active. Per NIST AU-3, ensure log records capture: event type, timestamp, source IP, username, and key fingerprint. No public IOC indicators (IPs, domains, hashes) are available for this vulnerability at time of disclosure.

Framework Mappings

MITRE-ATTACK

- **T1650** — Acquire Access
- **T1078** — Valid Accounts
- **T1556** — Modify Authentication Process

NIST-800-53R5

- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **AC-3** — Access Enforcement
- **SC-13** — Cryptographic Protection

OWASP-TOP10-2021

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

CIS-V8

- **6.1** — Establish an Access Granting Process
- **6.8** — Define and Maintain Role-Based Access Control
- **6.2** — Establish an Access Revoking Process
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

SOC2-TSC

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1650	Acquire Access	Resource-Development
T1078	Valid Accounts	Defense-Evasion
T1556	Modify Authentication Process	Credential-Access

Sources

Source	URL	Tier
MSRC Update Guide	https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-46595	T1
(consolidated)	https://api.msrc.microsoft.com/cvrf/v3.0/cvrf/2026-May	T1
CVE-2026-46595 | Mondoo Vulnerability Intelligence	https://mondoo.com/vulnerability-intelligence/vulnerability/CVE-202...	T3
CVE-2026-46595 - CVE Details, Severity, and Analysis | Strobes VI	https://vi.strobes.co/cve/CVE-2026-46595	T3
Linux Distros Unpatched Vulnerability : CVE-2026-46595 | Tenable®	https://www.tenable.com/plugins/nessus/316589	T3
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-46595	T1

Generated 2026-05-27 06:38 UTC by TJS Security Command Center