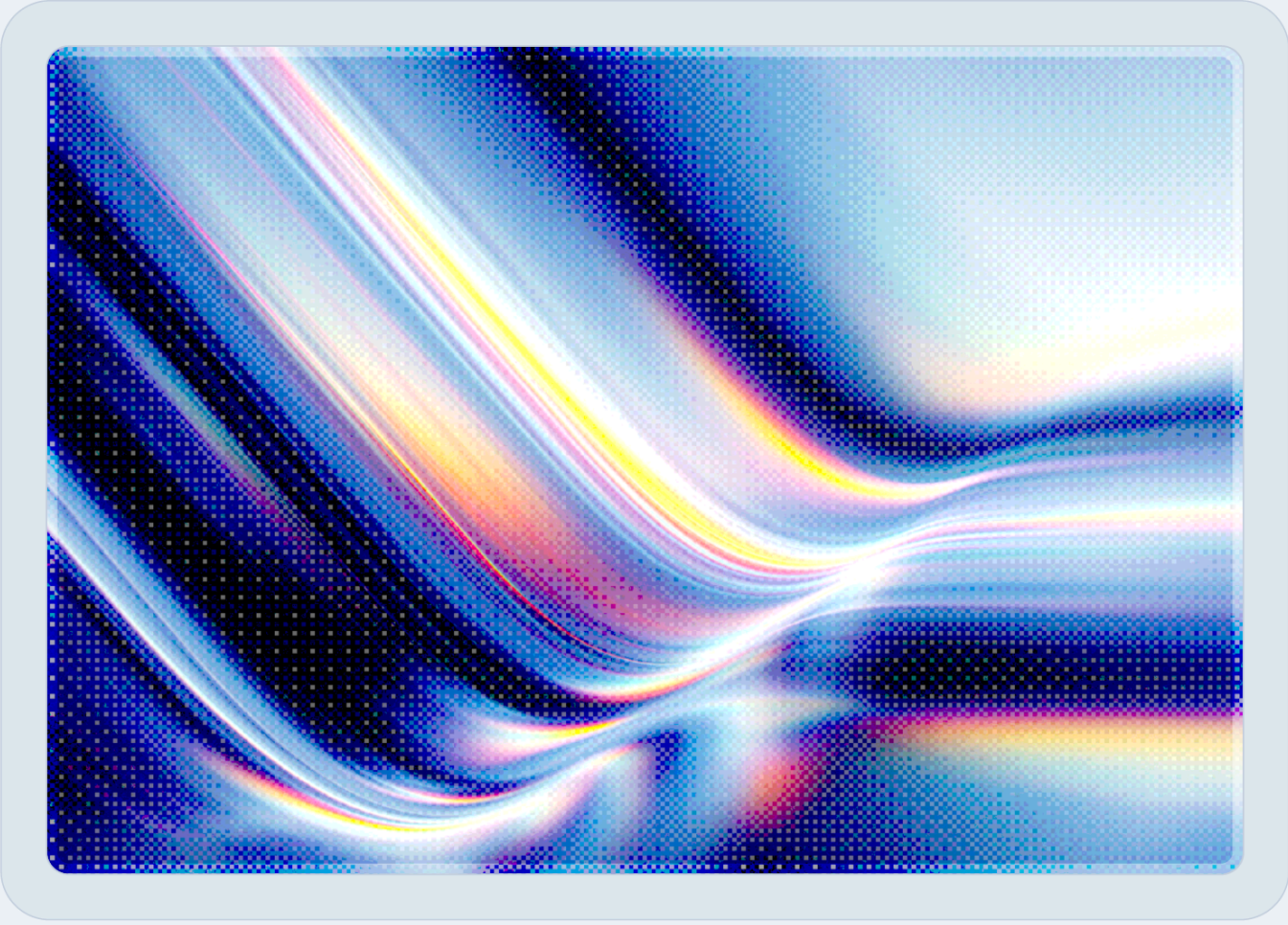
CSAI Foundation | Cloud Security Alliance

Trusted by Certificate, Not by Key: CVE-2026-112816 in LXD, MicroCloud and Incus

Security Implications and Guidance for Mutual TLS Client Impersonation in System Container and VM Platforms

2026-11-01

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

CVE-2026-112816 is an authentication bypass in the HTTPS API of Canonical LXD and MicroCloud, and the same flaw is present in the community fork Incus. The record carries a CVSS v4.0 base score of 10.0 and is classified as CWE-295, Improper Certificate Validation [1][2]. An attacker who can reach the API listener and who holds the *public* certificate of a trusted client can authenticate as that client without its private key. If the impersonated identity is privileged, the attacker may gain root on the host and full control of the cluster and its virtual machines [1][2].

The root cause is that the mutual TLS (mTLS) layer compared every certificate in the client-supplied chain against the trust store. In a TLS handshake, however, only the leaf certificate's private key is proven; the other certificates in the chain are attacker-controlled bytes [1][3].

Canonical's advisory of 28 October 2026 announced fixed snaps for LXD 6.10.1, 5.21.9, 5.0.11 and 4.0.15 and for MicroCloud 2.1.5 and 3.4.1, which reached stable channels on 29 October 2026 following a progressive rollout from candidate [1]. Incus fixed the issue in 7.6.0, published 30 October 2026 [3][4]. The Incus maintainers state that no configuration setting mitigates the flaw [3], and Canonical's advisory lists no workaround other than upgrading [1]. Until systems are patched, restricting network reachability of the HTTPS listener is the principal interim control.

As of this writing, CISA's SSVC assessment in the CVE record lists exploitation as "none," automatable as "yes" and technical impact as "total," and the vulnerability is not in the CISA Known Exploited Vulnerabilities catalog [2]. A repository claiming to be a proof of concept has nonetheless been indexed by a public exploit aggregator [5]. CSA has not verified it. Given the low technical barrier and the publicly indexed (unverified) proof of concept, defenders should plan as if exploitation is imminent.

Background

LXD is Canonical's manager for system containers and virtual machines. MicroCloud is Canonical's opinionated packaging of LXD with distributed storage and networking for small clusters. Incus is the community fork of LXD maintained under the Linux Containers project. All three expose a REST API over

HTTPS for remote management, which administrators commonly bind to a network address (TCP 8443 in the vendor documentation's examples) so that CLI clients, web UIs, automation pipelines and other cluster members can reach it [6].

The default authentication model for this API resembles SSH. Each client generates a key pair and a self-signed certificate. The server keeps a trust store of client certificates, and a client presenting a certificate found in that store is granted access with whatever permissions that store entry carries [6]. New clients are typically admitted with a one-time trust token, which adds their certificate to the trust store. Administrators can alternatively run in PKI mode, where a central certificate authority issues client and server certificates, or they can use OpenID Connect (OIDC) for human users [6].

The security of this model depends on a single property: the TLS handshake must prove that the party connecting holds the private key for the certificate the server matches against its trust store. In a standard TLS handshake (LXD's documentation requires TLS 1.3 [6]), the client sends a certificate chain and then a CertificateVerify message signed with the private key of the first (leaf) certificate. Any intermediate certificates that follow are there to help the server build a path to a trusted root. Nothing in the handshake proves that the client possesses keys for those additional certificates.

MicroCloud adds a cluster-formation step on top of this model. When a new machine joins, the initiator and joiner discover each other over mDNS and authenticate join messages with an HMAC keyed by an operator-supplied or generated passphrase. Only after that check does the initiator place the joiner's certificate in a temporary trust store [7][8]. The passphrase protects the integrity of the join exchange, but the joining member's public certificate still travels across the network during the process.

The disclosure unfolded over three days. Canonical published its advisory on 28 October 2026, noting that the vulnerability was referred to it by the Incus project and thanking Incus maintainer Stéphane Graber [1]. The Incus maintainers published a companion advisory (GHSA-xg75-xf37-cph9) on 29 October 2026, rated Critical with a CVSS v3.1 score of 10.0 [3]. The CVE record, CVE-2026-112816, assigned by Canonical as CNA, went public on 30 October 2026 [2]. The Incus advisory lists no CVE identifier of its own at the time of writing [3].

Security Analysis

Root Cause: Matching the Chain Instead of the Key

Canonical describes the defect directly: "Remote API authentication logic for mutual TLS compared all peer certificates sent by the client to known trust store entries," even though only the leaf certificate proves ownership of the signing key [1]. The Incus advisory supplies more detail. The HTTPS listener

requested client certificates but did not verify them as a chain, so the handshake only proved key possession for the first certificate. Authentication and certificate-ownership checks then iterated over the entire presented chain [3].

An attacker can therefore generate a throwaway key pair, place their own certificate in the leaf position, and append the public certificate of a trusted client as a fake "intermediate." The handshake succeeds because the attacker legitimately holds the leaf key. The application layer then finds the appended certificate in the trust store and treats the session as that trusted identity. The Incus maintainers state that the flaw has existed since the project's initial release [3]. Because Incus forked from LXD, this suggests the faulty pattern is inherited from shared code. The CVE record, however, lists only narrow version ranges immediately preceding each LXD fix [2]. Pending clarification from Canonical, operators should treat every LXD release earlier than the fixed versions as potentially affected.

The flaw falls squarely under CWE-295 [2]: the identity check accepted a certificate's presence in the chain as equivalent to proof of possession of its key. This distinction between holding a certificate and controlling its key is central to CSA's guidance on using asymmetric cryptography for zero trust, which treats cryptographic key control, rather than a copyable identifier, as the basis for authentication [9].

Why Public Certificates Are Not Secrets

The attack's precondition, possession of a trusted public certificate, is much weaker than it first appears. Public certificates are designed to be shared, and the Incus advisory enumerates several ways an attacker might obtain one [3]. Server certificates are returned in every TLS handshake, so any party that can connect to the listener can collect them without prior access. Trusted client certificates can be read by any trusted client through the certificates API, and they are also presented to every server a client connects to, so a malicious or compromised remote could harvest them. In MicroCloud, Canonical warns that an attacker who intercepts a joining member's public certificate during cluster formation can authenticate as that member [1].

The table below summarizes the identities an attacker could impersonate and the resulting impact, as described in the vendor advisories.

Impersonated identity	How the public certificate may be obtained	Reported impact
Privileged API client (admin)	Certificates API (as any trusted client); observing the client connect to an attacker-controlled remote	Full API control; possible root on host [1][3]

Impersonated identity	How the public certificate may be obtained	Reported impact
Cluster member / server certificate	Returned in every TLS handshake; intercepted during MicroCloud join	Internal cluster access, database gateway access, authorization bypass (Incus) [1][3]
Metrics client	Certificates API	Access scoped to metrics identity [3]
Host-to-VM agent connection	Not specified	Agent channel authentication affected (Incus) [3]
Token or OIDC client	Attacker presents arbitrary certificate during handshake	Influence over scriptlet-based authorization decisions (Incus) [3]

Beyond the identities in the table, the Incus advisory notes additional consequences. Attackers can renew or delete trust store entries through self-service paths. Because the unverified chain was also passed to the authorization scriptlet, token and OIDC clients could present chosen certificates to influence scriptlet-based authorization [3]. Canonical's advisory does not enumerate the server-certificate and scriptlet paths for LXD, so readers should not assume those Incus-specific findings map one-to-one onto LXD without vendor confirmation. Canonical's CVE description is nonetheless unambiguous that an unauthenticated attacker can authenticate as any trusted client and gain full access to the cluster and managed virtual machines [2].

Exposure and Blast Radius

The CVSS v4.0 vector assigned by Canonical (AV:N/AC:L/AT:N/PR:N/UI:N with high impact on both the vulnerable and subsequent systems) reflects a network-reachable, low-complexity, unauthenticated attack whose effects extend beyond the API process to the host and its workloads [2]. The Incus CVSS v3.1 vector uses a changed scope (S:C) for the same reason [3]. In practice, exposure depends on where the HTTPS listener is reachable. Deployments that bind the API only to a Unix socket are not remotely exposed through this path. Clusters, MicroCloud installations and any host managed remotely by tooling must listen on the network by design, however.

The blast radius matters particularly for AI and data workloads. LXD and Incus are used to host GPU-backed virtual machines, isolated build environments and sandboxes for untrusted code. In those settings, an administrative API identity is effectively a hypervisor credential. An attacker impersonating such an

identity could launch privileged instances, attach host devices, read instance storage, or alter the profiles that later workloads inherit. These are inferences about typical deployments rather than reported incidents, but they show why a "total" technical impact rating [2] should be taken literally.

Exploitation Status

CISA's SSVC enrichment of the CVE record, dated 30 October 2026, records exploitation as "none," automatable as "yes," and technical impact as "total" [2]. The EPSS probability shown in the OpenCVE record at the time of review was below one percent [2]. Exploitation probability scores for newly published vulnerabilities often lag reality, however. The technique requires only a standard TLS library capable of sending a custom client chain, and a repository purporting to be a proof of concept is already indexed publicly [5]. CSA has not tested that repository and makes no claim about its functionality. Taken together, the low technical barrier and public tooling claims justify planning on the assumption that reliable exploitation is within reach of a motivated attacker.

Recommendations

The guidance below moves from immediate remediation through post-patch verification to longer-term architectural lessons. Because the flaw has no configuration workaround, patching is the only complete fix, and the remaining measures either reduce exposure until patches are applied or help determine whether the vulnerability was used before they were.

Immediate Actions

Organizations running LXD or MicroCloud should refresh to the fixed snap revisions without delay (`sudo snap refresh lxd` and `sudo snap refresh microcloud`). They should then confirm that every host and cluster member reports LXD 6.10.1, 5.21.9, 5.0.11 or 4.0.15, and MicroCloud 2.1.5 or 3.4.1, as appropriate for its channel [1]. Canonical noted that the point releases were rolling out progressively from candidate to stable channels at the time of the advisory [1]. Operators who hold snaps or control refresh windows should therefore verify the installed revision rather than assuming an automatic refresh has occurred. Incus users should upgrade to 7.6.0 or later, which addresses this vulnerability along with a batch of other security fixes [3][4][10]. Users of distribution-packaged LXD or Incus should check their distribution's security tracker for backported fixes.

Where patching cannot happen immediately, the principal interim control is to restrict who can reach the HTTPS listener; the Incus maintainers state explicitly that no configuration setting mitigates the flaw [3]. Operators should bind `core.https_address` to a dedicated management interface rather than all interfaces, enforce host and network firewall rules that admit only known administrative hosts and cluster peers, and remove any direct internet exposure of the API, including exposure through port forwards or load balancers. Neither advisory identifies PKI mode as a mitigation [1][3], and rotating client certificates does not help because the newly issued public certificates are just as obtainable as the old ones. Operators should not rely on either as a substitute for patching.

Short-Term Mitigations

Patching closes the door but does not reveal whether anyone walked through it while it was open. After upgrading, review the trust store (`lxc config trust list` or `incus config trust list`) for entries that were added, renamed or had restrictions removed without a corresponding change record, and remove unexplained entries. Inspect lifecycle and audit logs from the exposure window for unexpected instance creation, profile or project changes, device attachments, image imports and configuration keys that weaken isolation (such as privileged-container or raw configuration settings). Organizations forwarding LXD or Incus events to a central log platform should search for API activity originating from source addresses outside the expected administrative set. For MicroCloud, confirm that every cluster member is expected, and treat any member that joined while an untrusted party could observe the network as warranting closer review.

Because an attacker on an unpatched system would have appeared as a legitimate trusted identity, absence of obvious anomalies is not conclusive. Where a host was network-exposed beyond a tightly controlled management segment, consider re-provisioning sensitive instances from known-good images. Re-keying identities after patching is also worth considering, as a hygiene step rather than a fix.

Strategic Considerations

CVE-2026-112816 is a reminder that mTLS is only as strong as the application code that interprets the handshake. Some control planes, including LXD and Incus before this fix, request client certificates without having the TLS library verify them as a chain, relying instead on custom verification logic so that they can implement trust-on-first-use or fingerprint pinning [3]. That design is legitimate, but it moves responsibility for path validation and proof-of-possession checks from a well-tested library into bespoke code. Security reviews and procurement questionnaires for infrastructure platforms should therefore ask specifically how client identity is derived from the TLS handshake. As a rule, only the leaf certificate, whose key the handshake verifies, should be mapped to an identity.

Architecturally, management APIs for hypervisor-class systems should not be the sole line of defense. CSA's zero trust guidance calls for layered controls: network-level isolation of management planes, per-request authorization with least privilege, and continuous verification of identity and posture [9][11]. Using restricted certificates or fine-grained authorization so that day-to-day automation does not hold full administrative rights could have reduced the impact of this flaw for some identities. Placing management listeners behind an authenticating proxy or software-defined perimeter would likely have reduced exposure for all of them. Finally, organizations that use LXD, MicroCloud or Incus to isolate AI agents, untrusted code or multi-tenant GPU workloads should classify the management API as a tier-zero asset and monitor it accordingly.

CSA Resource Alignment

Several CSA publications provide context for both the root cause of this vulnerability and the controls that limit its impact. *Using Asymmetric Cryptography to Help Achieve Zero Trust Objectives* [9] is the most directly relevant. It explains how public key infrastructure and certificate-based authentication underpin zero trust identity, with authentication resting on cryptographic control of private keys. CVE-2026-112816 is a concrete case of what happens when an implementation substitutes certificate *presence* for key *possession*. Organizations using that guide to design mTLS-based service identity should add explicit chain-validation and leaf-binding checks to their implementation reviews.

Zero Trust Guiding Principles [11] frames the strategic lesson. Implicit trust in a network location or a single credential check is the failure mode zero trust is meant to eliminate. Its principles of assuming breach, verifying explicitly and enforcing least privilege map directly onto the compensating controls described above: management-plane segmentation, restricted client identities and post-patch verification of trust stores.

For the platform layer, *Best Practices for Implementing a Secure Application Container Architecture* [12] from the Application Containers and Microservices Working Group addresses securing container orchestration and management interfaces. Although it focuses on application containers, its guidance on protecting the management plane, limiting administrative access and monitoring orchestrator activity applies equally to system-container and VM managers such as LXD and Incus.

Finally, the CSA AI Controls Matrix (AICM) v1.1 [13] provides the control vocabulary for tracking remediation. Its Threat & Vulnerability Management (TVM) domain covers timely patching of critical vulnerabilities, and its Identity & Access Management (IAM) domain covers strong authentication and

least-privilege access for privileged interfaces. Its Infrastructure & Virtualization Security (IVS) domain covers segmentation of management networks. These domains are directly relevant where LXD, MicroCloud or Incus host AI training, inference or agent-sandbox workloads.

References

[1] Canonical (M. Laing). "Security advisory: Critical vulnerability found in Canonical LXD and MicroCloud." Ubuntu Discourse, 28 October 2026.

[2] OpenCVE. "CVE-2026-112816: Improper certificate validation in the HTTPS listener in LXD." OpenCVE (mirroring MITRE/NVD record with CISA ADP enrichment), 30 October 2026.

[3] Linux Containers / Incus (S. Graber). "Authentication and authorization bypass through unverified client certificate chains (GHSA-xg75-xf37-cph9)." GitHub Security Advisories, 29 October 2026.

[4] Linux Containers / Incus. "Incus v7.6.0 release." GitHub, 30 October 2026.

[5] Sploitus. "dehasoft-tr/cve-2026-112816 – PoC exploit for CVE-2026-112816." Sploitus, accessed 1 November 2026.

[6] Canonical. "Remote API authentication." LXD Documentation, accessed 1 November 2026.

[7] Canonical. "How to add a cluster member to an existing MicroCloud." MicroCloud Documentation, accessed 1 November 2026.

[8] Canonical. "Explicit trust establishment mechanism for MicroCloud." Ubuntu Discourse, accessed 1 November 2026.

[9] Cloud Security Alliance. "Using Asymmetric Cryptography to Help Achieve Zero Trust Objectives." CSA Zero Trust Working Group, 2024.

[10] Mallory. "Critical LXD and MicroCloud TLS Authentication Bypass Enables Client Impersonation." Mallory, November 2026.

[11] Cloud Security Alliance. "Zero Trust Guiding Principles." CSA Zero Trust Working Group, 2023.

[12] Cloud Security Alliance. "Best Practices for Implementing a Secure Application Container Architecture." CSA Application Containers and Microservices Working Group, 2019.

[13] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, 2025.