Client-side encryption (SSE-C)

You can use objects encryption via S3 API, according to theAmazon SSE-C specification. This means that you need to provide an encryption/decryption key with each request to the object.

You can store the encryption key in Barbican, and provide it to the S3 client at runtime. Below we will provide more detailed explanation regarding how to use this in your workloads.

Requirements

This guide assumes familiarity with the following tools:

- `python-openstackclient` (with the `python-barbicanclient` plugin),
- `pwgen` ,
- `rclone` version 1.54 or later.

Create encryption details

According to SSE-C specification, in order to use server-side encryption, any S3 client needs to provide three pieces of information, which it includes in the request headers for each S3 request being made:

- Encryption algorithm: the only valid option here is AES256.
- Encryption key: a generated random key that we will store in Barbican. It should be valid AES key, which means that key length must be 32 bytes.
- Encryption key checksum: MD5 checksum of the encryption key. It’s used for integrity checks.

In order to generate encryption key and store it in Barbican, proceed as follows.

1. Generate secret:

```
secret_raw=$(pwgen 32 1)
```

2. Store secret in Barbican:

```
barbican_secret_url=$(openstack secret store --name objectSecret --algorithm aes --bit-length 256 --payload ${secret_raw} -f value -c 'Secret href')
```

3. Retrieve secret from Barbican:

```
secret=$(openstack secret get ${barbican_secret_url} -p -c Payload -f value)
```

Managing encrypted objects in S3 (with `rclone`)

SSE-C encryption has been implemented/fixed with version 1.54. Prior rclone versions won't work here.

1. Download and install the latest rclone for your distribution: https://rclone.org/downloads/
2. Create or retrieve your access key ID and secret key.
3. Create a configuration file named `~/.rclone.conf`, with the following content:

```
[cleura]
type = s3
provider = Ceph
env_auth = false
access_key_id = <access key id>
secret_access_key = <secret key>
endpoint = <region>.citycloud.com:8080
acl = private
sse_customer_algorithm = AES256
```

4. Create an S3 bucket:

```
rclone mkdir cleura:encrypted
```

5. Sync a directory to the S3 bucket, encrypting the files it contains on upload:

```
rclone sync ~/media/ cleura:encrypted \
  --s3-sse-customer-key=${secret}
```

6. Retrieve a file from S3 and decrypt it:

```
rclone copy cleura:encrypted/file.png \
  --s3-sse-customer-key=${secret}
```

For more examples on how to use rclone, please use its reference documentation: https://rclone.org/docs/#subcommands

Last update: 2022-05-11

Created: 2022-03-08

Authors: Florian Haas, Foad Lind