rgw - Bug #38700

silent corruption using SSE-C on multi-part upload to S3 with non-default part size

03/12/2019 01:39 PM - László van den Hoek

Status:	Resolved	**Start date:**	03/12/2019
Priority:	High	**Due date:**	
Assignee:	Casey Bodley	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour
Source:		**Reviewed:**	
Tags:		**Affected Versions:**	v13.2.4
Backport:	nautilus,mimic,luminous	**ceph-qa-suite:**	
Regression:	No	**Pull request ID:**	27130
Severity:	2 - major		

Description

Multi-part uploads to S3 on RGW silently cause corruption if you use a non-default chunk size, e.g. 5242881 bytes, which is the default plus one.

I ran into this using Alpakka S3, which can be used to upload streams of (beforehand) unknown size to S3. This triggers the bug because only a minimum chunk size can be specified, not an exact size.

This problem does not occur with AWS S3.

Steps to reproduce:

- install the AWS CLI as per https://docs.aws.amazon.com/cli/latest/userguide/install-linux.html:
  pip3 install awscli --upgrade --user

- configure AWS CLI - provide credentials and use region default:
  aws configure

- Create a SSE-C key. Corruption will occur consistently with any key, but different keys will yield distinct corruption patterns.
  dd if=/dev/zero of=/tmp/ssec.key bs=1 count=32

- Generate test data - 10MB of NUL bytes (0x00):
  dd if=/dev/zero of=/tmp/foo bs=10000 count=1000

- Upload and download the file - this will be a multi-part upload because the input file is larger than the default multi-part chunk size (5242880, i.e. 5 *1024*1024 = 80 * 2^16 bytes)
  aws s3 cp /tmp/foo s3://<bucket_name>/ --sse-c AES256 --sse-c-key fileb:///tmp/ssec.key --endpoint-url https://<rados gateway host>
  aws s3 cp s3://<bucket_name>/foo /tmp/bar --sse-c AES256 --sse-c-key fileb:///tmp/ssec.key --endpoint-url https://<rados gateway host>

- Verify file integrity - no surprises here:
  sha256sum /tmp/foo -> f5e02aa71e67f41d79023a128ca35bad86cf7b6656967bfe0884b3a3c4325eaf
  sha256sum /tmp/bar -> f5e02aa71e67f41d79023a128ca35bad86cf7b6656967bfe0884b3a3c4325eaf

Now, it gets interesting.

- Set the chunk size to a non-default value and repeat the process:
  aws configure set default.s3.multipart_chunksize 5242881
  aws s3 cp /tmp/foo s3://<bucket_name>/ --sse-c AES256 --sse-c-key fileb:///tmp/ssec.key --endpoint-url https://<rados gateway host>
  aws s3 cp s3://<bucket_name>/foo /tmp/bar --sse-c AES256 --sse-c-key fileb:///tmp/ssec.key --endpoint-url https://<rados gateway host>
- Now, the downloaded file is corrupt!
  sha256sum /tmp/foo -> f5e02aa71e67f41d79023a128ca35bad86cf7b6656967bfe0884b3a3c4325eaf
  sha256sum /tmp/bar -> 2351da404fde47c360d201cf77311afd1d5e8cbfb601a1db1cee0b8f82124554

Taking a closer look at the files:

- There are indeed 10M bytes in the file:
  cat foo | wc -c
- Filter out all NUL bytes and count the remainder; this outputs 0, confirming that the original foo contains only NUL bytes:
  tr -d '\000' < foo | wc -c

Then, from the corrupted bar file, we take the first *n* bytes, filter the NULs, and count how many remain.

- For the first *chunksize* bytes, there are 0, meaning the contents of the first chunk are not corrupted:
  head -c 5242881 bar | tr -d '\000' | wc -c
- Take one more byte, and the output changes to 1, meaning corruption starts at the part uploaded second:
  head -c 5242882 bar | tr -d '\000' | wc -c

Examining bar in a file editor shows that the remainder of the file is indeed garbage.

Uploading and downloading the file multiple times with a particular SSE-C key yields the exact same result, so this is a deterministic bug.

Related issues:

Copied to rgw - Backport #39068: nautilus: silent corruption using SSE-C on m...	**Resolved**
Copied to rgw - Backport #39069: mimic: silent corruption using SSE-C on mult...	**Resolved**
Copied to rgw - Backport #39070: luminous: silent corruption using SSE-C on m...	**Resolved**

History

#1 - 03/14/2019 05:44 PM - Casey Bodley

- Assignee set to Casey Bodley

- Priority changed from Normal to High

#2 - 03/18/2019 09:49 PM - Casey Bodley

A similar report coming from Dan Smith via [ceph-users] Rados Gateway using S3 Api does not store file correctly:

> The file is 92MB in size. I have stored files much larger and much smaller. If I store the file WITHOUT using the Customer Provided 256-bit AES key using Server Side encryption, the file stores and retrieves just fine (SHA256 hashes match).
>
> If I store the file USING the 256-bit AES key using Server Side encryption, the file stores without error, however, when I retrieve the file and compare the hash of the file I retrieve from ceph against the hash of the original file, the hashes differ.
>
> I am using the AWSSDK.S3 nuget package version 3.3.31.24, with ceph version "ceph version 12.2.10-551-gbb089269ea (bb089269ea0c1272294c6b9777123ac81662b6d2) luminous (stable)"

Perhaps this sanitized header is useful:

```
PUT [redacted]/delete-me?partNumber=18&uploadId=2~2dt4pYGY3vfKxBb9FcbVlAnbz_z3HTV HTTP/1.1
Expect: 100-continue
x-amz-server-side-encryption-customer-algorithm: AES256
x-amz-server-side-encryption-customer-key: [redacted]
x-amz-server-side-encryption-customer-key-MD5: [redacted]
User-Agent: aws-sdk-dotnet-coreclr/3.3.31.24 aws-sdk-dotnet-core/3.3.32.2 .NET_Core/4.6.27317.07 OS/Microsoft_
Windows_10.0.17763 ClientAsync TransferManager/MultipartUploadCommand
Host: [redacted]
X-Amz-Date: [redacted]
X-Amz-Decoded-Content-Length: 5242880
X-Amz-Content-SHA256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD
Authorization: [redacted]
Content-Length: 5248726
Content-Type: text/plain

14000;chunk-signature=[redacted]
[payload here]
```

#3 - 03/18/2019 09:54 PM - Casey Bodley

- Status changed from New to Verified

I'm able to reproduce the issue with our s3tests case test_encryption_sse_c_multipart_upload() by adding 1 to the 5M part size:

```
diff --git a/s3tests/functional/test_s3.py b/s3tests/functional/test_s3.py
index f2deb8e..a067ba4 100644
--- a/s3tests/functional/test_s3.py
+++ b/s3tests/functional/test_s3.py
@@ -8789,7 +8789,7 @@ def test_encryption_sse_c_multipart_upload():
         'x-amz-server-side-encryption-customer-key-md5': 'DWygnHRtgiJ77HCm+1rvHw==',
         'Content-Type': content_type
     }
-    (upload, data) = _multipart_upload_enc(bucket, key, objlen,
+    (upload, data) = _multipart_upload_enc(bucket, key, objlen, part_size=1+5*1024*1024,
                                           init_headers=enc_headers, part_headers=enc_headers,
                                           metadata={'foo': 'bar'})
     upload.complete_upload()
```

```
======================================================================

FAIL: s3tests.functional.test_s3.test_encryption_sse_c_multipart_upload
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/cbodley/s3-tests/virtualenv/lib/python2.7/site-packages/nose/case.py", line 197, in runTest
    self.test(*self.arg)
  File "/home/cbodley/s3-tests/s3tests/functional/test_s3.py", line 8807, in test_encryption_sse_c_multipart_u
pload
    eq(data, test_string)

AssertionError: 'dVZduWXPJyzErZCgbML[...]rDqRyNOPBIdnsm' != 'dVZduWXPJyzErZCgbML[...]\xc1tS\r\x04\nJ!'
```

#4 - 03/19/2019 10:22 AM - László van den Hoek

A similar report coming from Dan Smith via [ceph-users] Rados Gateway using S3 Api does not store file correctly

Link: http://lists.ceph.com/pipermail/ceph-users-ceph.com/2019-March/033833.html

#5 - 03/22/2019 01:53 PM - Nathan Cutler

- Backport set to nautilus,mimic,luminous

#6 - 03/22/2019 03:33 PM - Casey Bodley

- Status changed from Verified to Need Review

- Pull request ID set to 27130

I'm testing a fix for this at https://github.com/ceph/ceph/pull/27130. It looks like the issue is only on the decrypt side, so existing encrypted data is not corrupted and can be recovered.

#7 - 03/22/2019 04:14 PM - Casey Bodley

s3tests in https://github.com/ceph/s3-tests/pull/263 will also need backports

#8 - 03/28/2019 05:31 PM - Casey Bodley

- Status changed from Need Review to Testing

#9 - 04/01/2019 03:27 PM - Casey Bodley

- Status changed from Testing to Pending Backport

#10 - 04/01/2019 03:50 PM - Abhishek Lekshmanan

- Copied to Backport #39068: nautilus: silent corruption using SSE-C on multi-part upload to S3 with non-default part size added

#11 - 04/01/2019 03:50 PM - Abhishek Lekshmanan

- Copied to Backport #39069: mimic: silent corruption using SSE-C on multi-part upload to S3 with non-default part size added

#12 - 04/01/2019 03:50 PM - Abhishek Lekshmanan

- Copied to Backport #39070: luminous: silent corruption using SSE-C on multi-part upload to S3 with non-default part size added

#13 - 04/01/2019 04:54 PM - Casey Bodley

test cases in https://github.com/ceph/s3-tests/pull/266 can be backported as well

#14 - 04/08/2019 12:08 PM - Nathan Cutler

- Status changed from Pending Backport to Resolved