4 ActiveStorage direct upload fails to sign content-length header for S3 service

Share:

TIMELINE

travispew submitted a report to Ruby on Rails. Feb 5th (3 years ago)

When a user makes a direct upload using ActiveStorage, the browser makes a request to the DirectUploadsController containing the direct_upload parameters filename, content_type, byte_size, and checksum. These are used to generate a presigned url that is then passed back to the browser, allowing the user to upload directly to S3.

In particular, the byte_size parameter is intended to be encoded in the url for content-length, preventing the user from uploading a file of a different size. Although Rails does not currently provide any built in validations, developers have been encouraged to modify the controller or provide their own controller if they want to create a validation. For example, a developer might decide to prohibit uploads greater than 10MB in size.

in all current version of Rails with ActiveStorage and direct uploads `active_storage/lib/active_storage/service/s3_service.rb`, the code generates the presigned_url as follows:

Code 396 Bytes Wrap lines Copy Download

```
def url_for_direct_upload(key, expires_in:, content_type:, content_length:, checksum:)
  instrument :url, key: key do |payload|
    generated_url = object_for(key).presigned_url :put, expires_in: expires_in.to_i,
      content_type: content_type, content_length: content_length, content_md5: checksum

    payload[:url] = generated_url

    generated_url
  end
end
```

However, the aws-sdk-s3 gem *silently blacklists* the "content-length" header:

https://github.com/aws/aws-sdk-ruby/blob/master/gems/aws-sdk-s3/lib/aws-sdk-s3/presigner.rb#L22

This issue is also raised here: https://github.com/aws/aws-sdk-ruby/issues/2098

As a result, the content-length header is never actually part of the presigned url. As a result, a malicious user can select a file of arbitrary size, tell the direct uploads controller that the file is a different size, and then proceed to upload the file, bypassing the intended protection of the signed url.

The solution is to add the whitelist_headers argument:

Code 445 Bytes Wrap lines Copy Download

```
def url_for_direct_upload(key, expires_in:, content_type:, content_length:, checksum:)
  instrument :url, key: key do |payload|
    generated_url = object_for(key).presigned_url :put, expires_in: expires_in.to_i,
      content_type: content_type, content_length: content_length, content_md5: checksum,
      whitelist_headers: ['content-length']

    payload[:url] = generated_url

    generated_url
  end
end
```

After this is added, the content-length will be included in the presigned url and the client will be unable to upload a file of arbitrary size.

Impact

The attacker could upload a file of any size, unless the S3 service is configured separately to prevent this, whereas the developer believes they have protected themselves against this. This could allow an attacker to upload a very large file to S3, incurring additional costs to the website owner or causing other harm.

h1_analyst_caesar HackerOne triage changed the status to • Needs more info. Feb 6th (3 years ago)

Hello @travispew,

Thanks for your report.
In order to better assess the vulnerability, can you please provide a way to reproduce it?
For instance, can you create a small ruby web app with the vulnerable function and provide a video on how you are able to exploit the vulnerable code?

Best regards,
@turtle_shell

travispew changed the status to • New. Feb 6th (3 years ago)

Hi @turtle_shell ,

Sure! I've attached a very simple rails app that uses active storage with s3 direct upload as well as attached a video with how it can be exploited. By simply modifying the byte_size parameter sent to the directuploadscontroller, I was able to trick the controller into thinking the file was smaller than it was, and received a presigned url for S3 back. Because the content-length header is unexpectedly blacklisted and dropped by aws-sdk-s3, the presigned url generated by rails ActiveStorage doesn't actually use the provided content-length, and my upload of a larger than permitted file to S3 completes.

Had the active_storage code added `whitelist_headers: ["content-length"]` to the presigned_url method call, this file upload would not have been allowed by S3

Travis

2 attachments:
F708076: ActiveStorageValidation.mov
F708077: sample-activestorage-validation-app-master.zip

h1_analyst_caesar HackerOne triage updated the severity from Medium (4.3) to Medium (5.3). Feb 6th (3 years ago)

h1_analyst_caesar HackerOne triage changed the status to **Triaged**. Feb 6th (3 years ago)

Hello @travispew,

Thank you for your submission! We were able to validate your report, and have submitted it to the appropriate remediation team for review. They will let us know the final ruling on this report, and when/if a fix will be implemented. Please note that the status and severity are subject to change.

Regards,
@turtle_shell

travispew posted a comment. Updated Feb 6th (3 years ago)

Also, here is a failing test for the test suite for `test/service/s3_service_test.rb`:

Code 858 Bytes Wrap lines Copy Download

```
test "directly uploading file of different size" do
  key      = SecureRandom.base58(24)
  data     = "Something else entirely!"
  checksum = Digest::MD5.base64digest(data)
  url      = @service.url_for_direct_upload(key, expires_in: 5.minutes, content_type: "text/plain", content_length: data.size - 1, checksum: checksu

  uri = URI.parse url
  request = Net::HTTP::Put.new uri.request_uri
  request.body = data
  request.add_field "Content-Type", "text/plain"
  request.add_field "Content-MD5", checksum
  upload_result = Net::HTTP.start(uri.host, uri.port, use_ssl: true) do |http|
    http.request request
  end

  assert_equal upload_result.code, "403"
  assert_raises ActiveStorage::FileNotFoundError do
    @service.download(key)
  end
ensure
  @service.delete key
end
```

If the header is whitelisted, the test will pass as the upload is correctly rejected by S3

travispew posted a comment. Mar 7th (3 years ago)

Hi, trying to follow up about this report. Has anyone looked at it?

bluetooth_headset posted a comment. Mar 12th (3 years ago)

Hi @travispew - We are still looking into this report and will update you when we have additional information to share. Thanks for your patience!

tenderlove Ruby on Rails staff closed the report and changed the status to **Resolved**. May 18th (3 years ago)

Shipped

The Internet Bug Bounty rewarded travispew with a **$500** bounty. May 18th (3 years ago)

tenderlove Ruby on Rails staff requested to disclose this report. May 18th (3 years ago)

travispew agreed to disclose this report. May 18th (3 years ago)

This report has been disclosed. May 18th (3 years ago)