Arvados - Bug #17106

Cannot use federated tokens with keep-web S3 API

11/12/2020 04:46 PM - Peter Amstutz

Status:	Resolved	**Start date:**	11/20/2020
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2020-12-02 Sprint		

Description

Reported by user testing that they could use local tokens but not federated tokens to access the S3 API.

(Unclear if the "local" tokens that worked were bare tokens or v2 tokens)

Subtasks:

Task # 17142: Review 17106-s3-fed-token	**Resolved**

Associated revisions

Revision 917330c8 - 11/24/2020 04:55 PM - Tom Clegg

Merge branch '17106-s3-fed-token'

fixes #17106

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tom@tomclegg.ca>

History

#1 - 11/12/2020 04:46 PM - Peter Amstutz

- Status changed from New to In Progress

#2 - 11/12/2020 04:46 PM - Peter Amstutz

- Status changed from In Progress to New

- Subject changed from Cannot use federated tokens with keep-web S3 API to Cannot use federated tokens with keep-web S3 API

#3 - 11/12/2020 04:46 PM - Peter Amstutz

- Assigned To set to Tom Clegg

#4 - 11/12/2020 04:47 PM - Peter Amstutz

- Description updated

#5 - 11/12/2020 04:48 PM - Peter Amstutz

- Description updated

#6 - 11/16/2020 06:23 PM - Peter Amstutz

- Target version changed from 2020-12-02 Sprint to 2020-11-18

- Status changed from New to In Progress

#7 - 11/16/2020 06:28 PM - Peter Amstutz

- Release set to 36

#8 - 11/16/2020 10:16 PM - Peter Amstutz

- Release changed from 36 to 37

#9 - 11/17/2020 02:28 AM - Tom Clegg

Our preferred approach for s3 credentials (access key = token uuid, secret key = token secret part) can't work on a cluster that uses a remote LoginCluster, because keep-web can't use the uuid to look up the secret part (that only works if the token is in its own local database).

Using the entire v2 token as the access key and secret key is unlikely to work, because "/" is used as a delimiter in the authorization header, and real AWS access keys don't use that character. Some clients might handle this in various different ways, but some will surely reject it.

We can accept the entire v2 token as the access key and secret key if "/" is replaced with "_". 17106-s3-fed-token @ 70be08860db9e45d78a037d86b9a0420f1e392a1 -- https://ci.arvados.org/view/Developer/job/developer-run-tests/2177/

Using just the secret part should be possible, but doesn't currently work. I suspect we need to change RailsAPI's verification process so it calls "get current token" to get the original UUID (it currently only calls "get current user" to verify that the token is valid, then invents a UUID for the local database).

#10 - 11/18/2020 02:40 PM - Tom Clegg

17106-s3-fed-token @ f194bb8b667815f3f3fbd01a3d7ba824b05416ed -- https://ci.arvados.org/view/Developer/job/developer-run-tests/2182/

#11 - 11/18/2020 03:30 PM - Tom Clegg

17106-s3-fed-token @ bee9aff3bd6b69f81a0dd53fa7b4118d0eeeb0a9

- Accept secret part of token, even if token was issued by LoginCluster
- Accept full v2 token, with "/" replaced by "_" (unsure whether we want to keep/document this; it's awkward but it can make more federation cases work)
- Comments/docs updated since test run above in note-10

#12 - 11/18/2020 04:29 PM - Tom Clegg

- Target version changed from 2020-11-18 to 2020-12-02 Sprint

#13 - 11/19/2020 10:57 PM - Peter Amstutz

Tom Clegg wrote:

> 17106-s3-fed-token @ bee9aff3bd6b69f81a0dd53fa7b4118d0eeeb0a9
>
> - Accept secret part of token, even if token was issued by LoginCluster

So if you can present a bare secret and if the LoginCluster accepts it, it can be used. It behaves the same way whether it is an OIDC access token or just a bare arvados token? Meaning what gets stored is the hash?

So in both the OIDC access token and bare token cases, you have to present "secret/secret" (or secret/none). Does the value of the SecretKey even matter in this case, except to be able to check if the signature is correct or not, even though we are not basing out authentication decision on the signature?

> - Accept full v2 token, with "/" replaced by "_" (unsure whether we want to keep/document this; it's awkward but it can make more federation cases work)

I think the federation case you are referring to is this: you can present the munged v2 token in the "access key" part and then it gets decoded to a regular v2 token by keep-web, now we have a token uuid telling us what cluster issued the token, for the case where you have a federation but not a LoginCluster.

I don't suppose we could work around this by supporting AWS signatures in the API server? Or does that not work because the signature is an hmac that includes the actual request, so we'd have to somehow encapsulate/forward the entire request (more work, might be awkward)?

So I think I understand the thought process that gets us here. The underscore workaround seems fine and should stay. Providing access instructions in Workbench (#16622) we can provide the modified token for copy and paste.

#14 - 11/20/2020 03:26 PM - Tom Clegg

Peter Amstutz wrote:

> So if you can present a bare secret and if the LoginCluster accepts it, it can be used. It behaves the same way whether it is an OIDC access token or just a bare arvados token? Meaning what gets stored is the hash?

Yes, exactly.

> So in both the OIDC access token and bare token cases, you have to present "secret/secret" (or secret/none). Does the value of the SecretKey even matter in this case, except to be able to check if the signature is correct or not, even though we are not basing out authentication decision on the signature?

For v4 requests we do check the signature, so "none" wouldn't work.

For v2 requests we don't check the signature, we just check that the access key is a real token.

Supporting v2 signatures might not be important enough to bother implementing a proper signature check. Perhaps we should just stop accepting them, like AWS.

> - Accept full v2 token, with "/" replaced by "_" (unsure whether we want to keep/document this; it's awkward but it can make more federation cases work)
>
> I think the federation case you are referring to is this: you can present the munged v2 token in the "access key" part and then it gets decoded to a regular v2 token by keep-web, now we have a token uuid telling us what cluster issued the token, for the case where you have a federation but not a LoginCluster.

Yes, exactly.

> I don't suppose we could work around this by supporting AWS signatures in the API server? Or does that not work because the signature is an hmac that includes the actual request, so we'd have to somehow encapsulate/forward the entire request (more work, might be awkward)?

We could give controller a "validate v4 signature" endpoint, so an intermediate cluster could accept a request that uses only the token UUID as its access key. I think that part would be easy enough. However, the intermediate cluster still wouldn't know the entire token, so it wouldn't be able to retrieve/update collections needed to *serve* the request. (Even in the cases where the "validate v4 signature" endpoint could technically look up and return the secret, that doesn't seem like a good road to travel.)

> So I think I understand the thought process that gets us here. The underscore workaround seems fine and should stay. Providing access instructions in Workbench (#16622) we can provide the modified token for copy and paste.

Good point, that goes a long way to addressing the UX weirdness.

#15 - 11/20/2020 03:48 PM - Tom Clegg

17106-s3-fed-token @ 199ca290ab259ba21f798bb059bb808fe3b609ba -- https://ci.arvados.org/view/Developer/job/developer-run-tests/2188/

- Updates docs re "_" substitution
- Accepts "_" substitution with v2-signed requests too, to simplify usage/docs

One more edge that we will have to confront eventually (but doesn't affect the present issue/branch):

If the token is an OIDC access token rather than an Arvados token, it might contain both "/" and "_", so this substitution won't work. As the code stands now, an OIDC access token containing "_" will work (provided it doesn't happen to start with "v2_" like an arvados token), but an OIDC access token with "/" that the user has replaced with "_" will not work.

OIDC access tokens are allowed to have any printable char, so I think handling the general case would require escaping ("=2f"?) rather than substituting.

#16 - 11/20/2020 05:01 PM - Peter Amstutz

Tom Clegg wrote:

> 17106-s3-fed-token @ 199ca290ab259ba21f798bb059bb808fe3b609ba -- https://ci.arvados.org/view/Developer/job/developer-run-tests/2188/
>
> One more edge: if the token is an OIDC access token rather than an Arvados token, it might contain both "/" and "_", so this substitution won't work. As the code stands now, an OIDC access token containing "_" will work (provided it doesn't happen to start with "v2_" like an arvados token), but an OIDC access token with "/" that the user has replaced with "_" will not work.
>
> OIDC access tokens are allowed to have any printable char, so I think handling the general case would require escaping ("=2f"?) rather than substituting.

So as a general solution instead of substituting "_" for "/" we do URI-encode or other escaping? Seems like if we are going to change the substitution strategy we should do this now, otherwise we merge support for the underscore syntax only to pull it out again (or be stuck with it forever).

#17 - 11/20/2020 06:54 PM - Tom Clegg

I was thinking we would continue to accept the "_" substitution for Arvados tokens ("v2_zzzzz-gj3su-..." is recognizable) even when we also accept a more general escaping mechanism for non-Arvados tokens.

#18 - 11/20/2020 07:00 PM - Peter Amstutz

Tom Clegg wrote:

> I was thinking we would continue to accept the "_" substitution for Arvados tokens ("v2_zzzzz-gj3su-..." is recognizable) even when we also accept a more general escaping mechanism for non-Arvados tokens.

I don't understand the benefit if we end up with two schemes and there's some ambiguity if "_" should be turned back into "/" or not.

#19 - 11/23/2020 07:59 PM - Tom Clegg

I just meant a string like "v2_{uuid}_..." will still be unambiguous if/when we also support a more general escaping scheme, so we won't need to worry about accidentally mangling "_" chars in tokens that aren't Arvados tokens.

Added support for query-escaped tokens.

17106-s3-fed-token @ 0cfb2b0646ad8129c82883717af7a51d28e6876a -- https://ci.arvados.org/view/Developer/job/developer-run-tests/2193/

#20 - 11/23/2020 08:53 PM - Peter Amstutz

Tom Clegg wrote:

> I just meant a string like "v2_{uuid}_..." will still be unambiguous if/when we also support a more general escaping scheme, so we won't need to worry about accidentally mangling "_" chars in tokens that aren't Arvados tokens.
>
> Added support for query-escaped tokens.
>
> 17106-s3-fed-token @ 0cfb2b0646ad8129c82883717af7a51d28e6876a -- https://ci.arvados.org/view/Developer/job/developer-run-tests/2193/

1. I think you want to use "url.PathUnescape" not "url.QueryUnescape" because "QueryUnescape" treats "+" as a space and "PathEscape" does not.

2. The documentation explaining tokens with S3 would benefit from an example:

"If have an Arvados token 'v2/zzzzz-gj3su-1234567890abcde/zyxzyxzyxzyx' you can use the following to communicate with the cluster 'zzzzz':

access_key = zzzzz-gj3su-1234567890abcde
secret_key = zyxzyxzyxzyx

however if this is a federated token and you are communicating with a cluster other than 'zzzzz', use this:

access_key = v2_zzzzz-gj3su-1234567890abcde_zyxzyxzyxzyx
secret_key = v2_zzzzz-gj3su-1234567890abcde_zyxzyxzyxzyx"

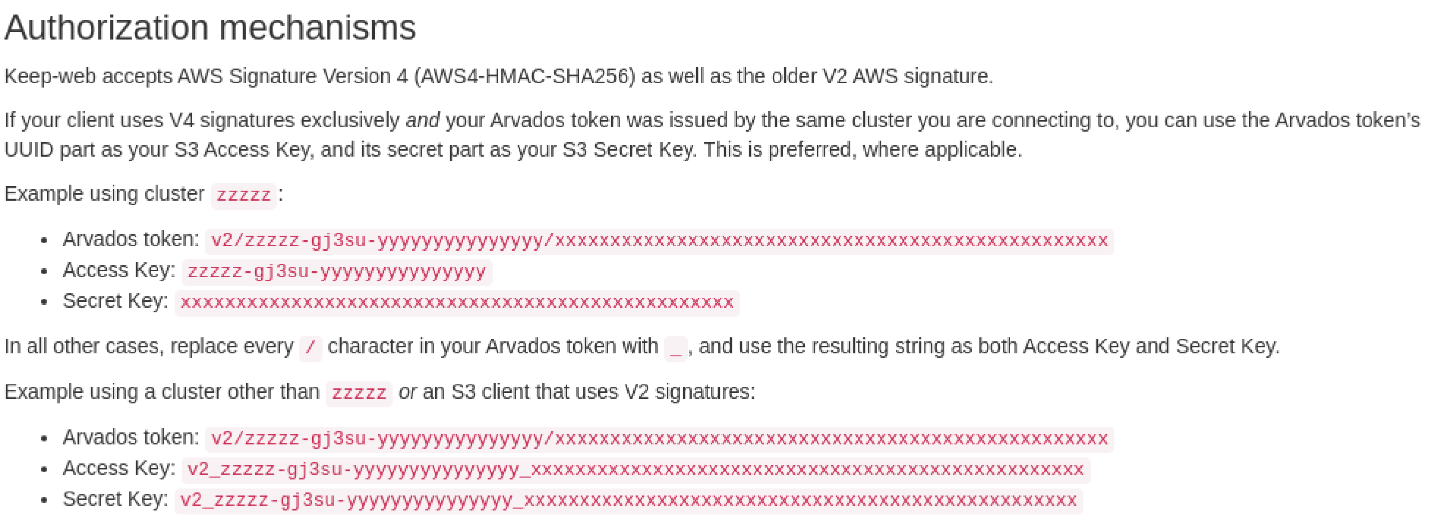
Rest LGTM.

#21 - 11/24/2020 03:29 PM - Tom Clegg

17106-s3-fed-token @ c36bac7d8ec9f7f579ddfdc06a328fa3668e80a3 -- https://ci.arvados.org/view/Developer/job/developer-run-tests/2195/

#22 - 11/24/2020 03:47 PM - Tom Clegg

- File doc.png added



Authorization mechanisms

Keep-web accepts AWS Signature Version 4 (AWS4-HMAC-SHA256) as well as the older V2 AWS signature.

If your client uses V4 signatures exclusively *and* your Arvados token was issued by the same cluster you are connecting to, you can use the Arvados token's UUID part as your S3 Access Key, and its secret part as your S3 Secret Key. This is preferred, where applicable.

Example using cluster `zzzzz`:

- Arvados token: `v2/zzzzz-gj3su-yyyyyyyyyyyyyyy/xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`
- Access Key: `zzzzz-gj3su-yyyyyyyyyyyyyyy`
- Secret Key: `xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`

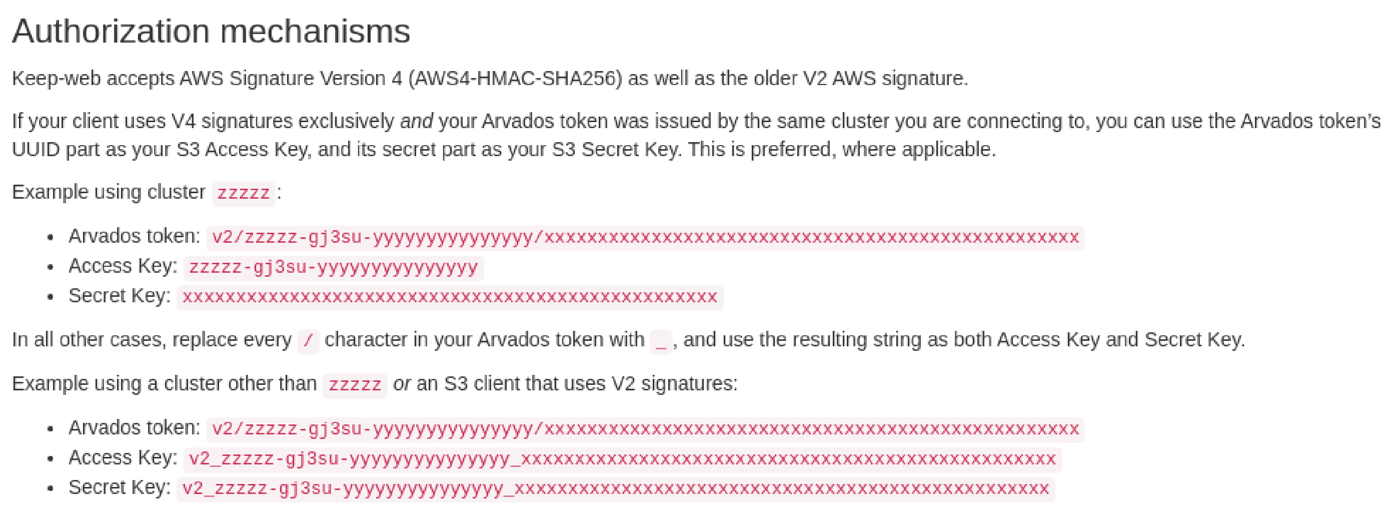
In all other cases, replace every `/` character in your Arvados token with `_`, and use the resulting string as both Access Key and Secret Key.

Example using a cluster other than `zzzzz` *or* an S3 client that uses V2 signatures:

- Arvados token: `v2/zzzzz-gj3su-yyyyyyyyyyyyyyy/xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`
- Access Key: `v2_zzzzz-gj3su-yyyyyyyyyyyyyyy_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`
- Secret Key: `v2_zzzzz-gj3su-yyyyyyyyyyyyyyy_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`

#23 - 11/24/2020 04:42 PM - Peter Amstutz

Tom Clegg wrote:



Authorization mechanisms

Keep-web accepts AWS Signature Version 4 (AWS4-HMAC-SHA256) as well as the older V2 AWS signature.

If your client uses V4 signatures exclusively *and* your Arvados token was issued by the same cluster you are connecting to, you can use the Arvados token's UUID part as your S3 Access Key, and its secret part as your S3 Secret Key. This is preferred, where applicable.

Example using cluster `zzzzz`:

- Arvados token: `v2/zzzzz-gj3su-yyyyyyyyyyyyyyy/xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`
- Access Key: `zzzzz-gj3su-yyyyyyyyyyyyyyy`
- Secret Key: `xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`

In all other cases, replace every `/` character in your Arvados token with `_`, and use the resulting string as both Access Key and Secret Key.

Example using a cluster other than `zzzzz` *or* an S3 client that uses V2 signatures:

- Arvados token: `v2/zzzzz-gj3su-yyyyyyyyyyyyyyy/xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`
- Access Key: `v2_zzzzz-gj3su-yyyyyyyyyyyyyyy_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`
- Secret Key: `v2_zzzzz-gj3su-yyyyyyyyyyyyyyy_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`

LGTM

#24 - 11/24/2020 05:05 PM - Anonymous

- Status changed from In Progress to Resolved

Applied in changeset arvados|917330c81bb370225ccd0e051dbdca3d1870710e.

#25 - 02/18/2021 10:52 PM - Peter Amstutz

- Release changed from 37 to 38

Files

doc.png	71.3 KB	11/24/2020	Tom Clegg