Arvados - Bug #8079

[API] Add uuid property to ApiClientAuthorization

12/23/2015 02:44 PM - Tom Clegg

Status:	Resolved	**Start date:**	02/12/2016
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:	API	**Estimated time:**	0.00 hour
Target version:	2016-03-16 sprint		

Description

This is more consistent with other objects, and makes it possible to specify an API token without revealing its secret content. For example, "arv edit {uuid}" can be used by an admin to change a token's scope.

Ensure it is not possible to retrieve an API token by looking up its UUID. Currently the filters behavior is altered such that ["uuid","=",api_token] looks up a token. This should change such that

- list?filters=[["uuid","=",X]] and get?uuid=X return the auth record with UUID X only if token X's API token is the current API token
- list?filters=[["api_token","=",T]] returns the auth record with the given api_token, but only if it belongs to the current user

Subtasks:

Task # 8366: Review branch: 8079-api-client-auth-uuid	**Resolved**
Task # 8661: Review 8079-lookup-token-uuid	**Resolved**

Related issues:

Blocks Arvados - Feature #8128: [Crunch2] API support for crunch-dispatch	**Resolved**	**04/28/2016**

Associated revisions

Revision f341c6a2 - 02/16/2016 02:32 AM - Radhika Chippada

closes #8079
Merge branch '8079-api-client-auth-uuid'

Revision e3c2f870 - 03/14/2016 07:06 PM - Tom Clegg

Merge branch '8079-lookup-token-uuid' closes #8079

History

#1 - 12/23/2015 02:48 PM - Tom Clegg

- Description updated

- Category set to API

#2 - 12/30/2015 11:55 PM - Brett Smith

- Target version set to Arvados Future Sprints

#3 - 01/22/2016 05:37 AM - Tom Clegg

- Description updated

#4 - 02/02/2016 08:00 PM - Tom Clegg

- Target version deleted (Arvados Future Sprints)

- Release set to 11

#5 - 02/03/2016 08:33 PM - Tom Clegg

- Target version set to 2016-02-17 Sprint

#6 - 02/03/2016 08:33 PM - Tom Clegg

- Story points changed from 0.5 to 1.0

#7 - 02/03/2016 08:34 PM - Radhika Chippada

- Assigned To set to Radhika Chippada

#8 - 02/09/2016 05:19 PM - Radhika Chippada

- Status changed from New to In Progress

#9 - 02/12/2016 07:49 PM - Nico César

reviewing 2d6884f5b20c349e7fd28a51cd876d40524186ad

small changes: services/api/db/structure.sql has a extra space after "Tablespace:"

I eyeball it the code and so far nothing pops ups as terrible

I'm running the tests now

#10 - 02/16/2016 02:35 AM - Radhika Chippada

- Status changed from In Progress to Resolved

- % Done changed from 0 to 100

Applied in changeset arvados|commit:f341c6a22325ab4de54a10c7262e3975fad36851.

#11 - 03/07/2016 06:28 PM - Tom Clegg

- Status changed from Resolved to In Progress

- Assigned To changed from Radhika Chippada to Tom Clegg

- Target version changed from 2016-02-17 Sprint to 2016-03-16 sprint

- Story points changed from 1.0 to 0.5

Story description asked for:

- list?filters=[["uuid","=",X]] and get?uuid=X return the auth record with UUID X only if token X's API token is the current API token

Current behavior (master @ be191fe) is:

- list?filters=[["uuid","=",X]] and get?uuid=X return the auth record with UUID X

When I changed the code to do what the story description asked, Workbench tests revealed that the "unshare" button needs to delete a collection-scoped token, which is impossible with the above spec, so I implemented:

- list?filters=[["uuid","=",X]] and get?uuid=X return the auth record with UUID X only if token X's API token is the current API token, **or token X belongs to the current user, and the current user is acting through a trusted client.**

8079-lookup-token-uuid @ afe4660

https://ci.curoverse.com/job/developer-test-job/93/

#12 - 03/09/2016 02:03 PM - Brett Smith

Reviewing afe4660. All the new code looks good. I'm running tests now, but everything's passed so far. At most, I have one question about it. I also have some questions about some existing code and how it relates to these change. If you think these are out of scope, that's totally fair, but I wanted to raise these in the hopes of avoiding another go-around on this ticket if we can help it.

In particular, there seems to be imperfect overlap in the security measures implemented by find_objects_for_index and current_api_client_is_trusted. Both seem to be aiming toward the same basic goal, but there are discrepancies in what they enforce, and how.

- find_objects_for_index scrubs @filters so it only includes exact matches on uuid and api_token.
- It also scrubs @where, but it allows any search on uuid. (This seems like a bug.)
- If I'm reading [['uuid'], ['api_token']].include? filters right, current_api_client_is_trusted returns an error to the client if it tries to filter on any other attribute. It doesn't check the operator.

This seems especially confusing because what happens depends on the order these filters run in. I believe find_objects_for_index runs first, since it's declared first. If that's right, I believe the condition in current_api_client_is_trusted is a noop, because find_objects_for_index will have already scrubbed @filters by the time that check runs.

I think I personally would advocate that we should commit to a single security strategy—either scrub filters, or fully reject requests with unallowed filters—and we should enforce it in one place. Having a little of both seems like an opportunity for corner cases to fall through the cracks.

If you decide all that's out of scope, my one question: would if [['uuid'], ['api_token']].include? filters be better written if (filters - %w(uuid api_token)).empty? ? It should be allowed to filter on both attributes simultaneously, right?

#13 - 03/12/2016 11:48 PM - Tom Clegg

Brett Smith wrote:

> Reviewing afe4660. All the new code looks good. I'm running tests now, but everything's passed so far. At most, I have one question about it. I also have some questions about some existing code and how it relates to these change. If you think these are out of scope, that's totally fair, but I wanted to raise these in the hopes of avoiding another go-around on this ticket if we can help it.

I agree this area seems more complicated than it deserves to be and should get cleaned up. I'm mostly inclined to do that in a follow-up branch rather than block the bugfix behind it (either on this issue# or another). OTOH, I would like to avoid adding new bugs while fixing this one, and if we need cleanup to make the code human-readable enough for a review, I'm for it.

- find_objects_for_index scrubs @filters so it only includes exact matches on uuid and api_token.
- It also scrubs @where, but it allows any search on uuid. (This seems like a bug.)

Everything in @where is implicitly an equality operator, so "any search" is still just exact matches, right?

- If I'm reading [['uuid'], ['api_token']].include? filters right, current_api_client_is_trusted returns an error to the client if it tries to filter on any other attribute. It doesn't check the operator.

> This seems especially confusing because what happens depends on the order these filters run in. I believe find_objects_for_index runs first, since it's declared first. If that's right, I believe the condition in current_api_client_is_trusted is a noop, because find_objects_for_index will have already scrubbed @filters by the time that check runs.

This condition is really a check that the results *are* filtered by uuid/token (not a check that they *aren't* filtered on any other field), for two reasons:

- close an information leak: without it, a query for "scopes=[...]" could result in either 403 or "only the token you're using", depending on whether any *other* tokens exist with that same set of scopes.
- cater to Workbench's current way of deciding whether to offer "Share" buttons, i.e., "does filtering on scopes return 403?". Until we teach Workbench a better way, we have to return 403 when the sharing token is used to do an index query that returns only itself.

...so I've changed the code to "filters must include at least one of (uuid, api_token)" and added comments to explain.

Also de-duplicated the current_api_client_is_trusted code so it just looks at @objects regardless of the current action.

Now at ffd4f23

#14 - 03/14/2016 02:02 PM - Brett Smith

Tom Clegg wrote:

> Brett Smith wrote:
>
> > Reviewing afe4660. All the new code looks good. I'm running tests now, but everything's passed so far. At most, I have one question about it. I also have some questions about some existing code and how it relates to these change. If you think these are out of scope, that's totally fair, but I wanted to raise these in the hopes of avoiding another go-around on this ticket if we can help it.
>
> I agree this area seems more complicated than it deserves to be and should get cleaned up. I'm mostly inclined to do that in a follow-up branch rather than block the bugfix behind it (either on this issue# or another). OTOH, I would like to avoid adding new bugs while fixing this one, and if we need cleanup to make the code human-readable enough for a review, I'm for it.

No, readability per se was never my concern. Addressing those concerns separately is fine.

> - It also scrubs @where, but it allows any search on uuid. (This seems like a bug.)
>
> Everything in @where is implicitly an equality operator, so "any search" is still just exact matches, right?

If I'm reading apply_where_limit_order_pararms correctly, you can use a where parameter to do a LIKE search by passing in a value ["contains", substring]. So, I don't think so.

Separately, shouldn't this limitation allow api_token through, the same way we consistently do with filters?

Everything else makes sense and looks good.

#15 - 03/14/2016 02:36 PM - Tom Clegg

Brett Smith wrote:

> If I'm reading apply_where_limit_order_pararms correctly, you can use a where parameter to do a LIKE search by passing in a value ["contains",

substring]. So, I don't think so.

Separately, shouldn't this limitation allow api_token through, the same way we consistently do with filters?

Indeed. Somehow I missed that "contains" introduction. Well, I'd prefer to remove that feature (we can do that stuff with filters now without the ambiguous syntax) but this has already crept far enough so I've settled for making the "where" check equivalent to the "filters" check (including the api_token discrepancy) like this:

```
      @where.select! { |attr, val|
        val.is_a?(String) && (attr == 'uuid' || attr == 'api_token')
      }
```

→ 11fc089

#16 - 03/14/2016 06:32 PM - Brett Smith

A comment about the purpose of val.is_a?(String) might be helpful to future readers. Please go ahead and merge. Thanks.

#17 - 03/14/2016 07:20 PM - Tom Clegg

- Status changed from In Progress to Resolved

- % Done changed from 50 to 100

Applied in changeset arvados|commit:e3c2f8706634a3e1e6e4ecc485dadbf5e82dd2a7.