Arvados - Feature #14262

[Controller] Specify runtime_token when creating container requests on a remote cluster

09/26/2018 04:05 PM - Tom Clegg

Status:	Resolved	**Start date:**	10/24/2018
Priority:	Normal	**Due date:**	
Assigned To:	Peter Amstutz	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2018-11-14 Sprint		

Description

When a client asks its home cluster to create a CR on a remote cluster, the resulting container should have privileges -- but the remote cluster cannot create a runtime token by itself that is capable of reading collections from any other cluster. Instead, the home cluster, when proxying the request, should use the runtime_token field to supply a token for the container to use.

When proxying a "create container request" request to a remote cluster, the controller should:

- Check whether the current token is a local token with scopes=["all"] (if not, skip the rest)
- Check whether a runtime_token was already supplied by the caller (if so, skip the rest)
- Create a new api_client_authorizations record, with an expiry time set to (now + blob_signature_ttl), and use that as runtime_token in the proxied request

Subtasks:

Task # 14300: Review 14262-remote-container	**Resolved**

Related issues:

Related to Arvados - Feature #14260: [API] Add "runtime_token" field to conta...	**Resolved**	**10/15/2018**
Related to Arvados - Feature #14200: [API] Reduce privilege exposure via API ...	**New**	

Associated revisions

Revision 12495b9c - 11/01/2018 07:57 PM - Peter Amstutz

Merge branch '14262-remote-container' refs #14262

Arvados-DCO-1.1-Signed-off-by: Peter Amstutz <pamstutz@veritasgenetics.com>

History

#1 - 09/26/2018 04:16 PM - Tom Clegg

- Related to Feature #14260: [API] Add "runtime_token" field to container_requests added

#2 - 09/26/2018 04:48 PM - Tom Clegg

- Story points set to 2.0

#3 - 10/01/2018 02:15 PM - Peter Amstutz

- Related to Feature #14200: [API] Reduce privilege exposure via API tokens in multi-cluster workflows added

#4 - 10/03/2018 03:38 PM - Peter Amstutz

- Assigned To set to Peter Amstutz

- Target version set to 2018-10-17 sprint

#5 - 10/03/2018 03:42 PM - Peter Amstutz

- Target version changed from 2018-10-17 sprint to Arvados Future Sprints

#7 - 10/03/2018 04:12 PM - Peter Amstutz

- Target version changed from Arvados Future Sprints to 2018-10-17 sprint

#8 - 10/17/2018 03:17 PM - Peter Amstutz

- Status changed from New to In Progress

- Target version changed from 2018-10-17 sprint to 2018-10-31 sprint

#9 - 10/22/2018 05:19 PM - Peter Amstutz

14262-remote-container @ 79e1faf117ec667aef01247fe1fe79f6588753c0

#10 - 10/24/2018 02:12 PM - Peter Amstutz

https://ci.curoverse.com/view/Developer/job/developer-run-tests/950/

#11 - 10/24/2018 07:58 PM - Peter Amstutz

14262-remote-container @ e92125d70c4b9b5fcaaee887942d415d4b197753

https://ci.curoverse.com/view/Developer/job/developer-run-tests/951/

- Use container runtime token to load Docker image, required to load container image from a remote cluster, and related refactoring / test fixes
- Reorganize controller federation code into a few different files
- API server accepts bare PDH for container_image if it wasn't explicitly rejected (defined as being readable on the local cluster but not returned by Collection.for_latest_docker_image
- Refactor validating API tokens in controller, add createAPIToken() which inserts token record by interacting directly with the database.
- Refactor controller proxy to separate ForwardRequest and ForwardResponse methods (replaces calling proxy.Do() with a filter)
- Refactor arvados.Client.RequestAndDecode() to split out client.MakeRequest() (intended to go back and update some of the federation methods to use MakeRequest() instead of constructing requests manually but that didn't make it into this branch.)

#12 - 10/26/2018 08:48 PM - Tom Clegg

In services/api/app/models/container.rb, the added "accept unreadable PDH as docker image" code seems like it belongs in models/collection.rb with the rest of the "choose PDH for docker image" code -- it seems like adding "if coll_match.empty? then return loc.to_s" there would avoid the duplicate code and database query.

This change means a bad/unreadable docker image PDH isn't detected until a worker VM comes up and tries to load it, as the deleted test case indicates. Controller could catch this by using searchRemoteClusterForPDH before passing along the "create CR" req. I think it's OK to accept the regression for the time being on sites that actually use federation, but it seems like we might as well retain the previous Rails behavior on sites that don't have any remotes configured.

Why rename (*proxy)Do (analogous to (*http.Client)Do()) to ForwardRequest?

In Do/ForwardRequest, why remove defer cancel()? Docs say we should call cancel.

The comments on ForwardResponse and ForwardRequest seem backwards: doesn't "downstream" normally mean "closer to client"?

In federation_test.go should the last two of the four new lines be swapped?

```
 func (s *FederationSuite) remoteMockHandler(w http.ResponseWriter, req *http.Request) {
+       b := &bytes.Buffer{}
+       io.Copy(b, req.Body)
+       req.Body = ioutil.NopCloser(b)
+       req.Body.Close()
        s.remoteMockRequests = append(s.remoteMockRequests, *req)
 }
```

The "shared mutex and pointers" approach in searchRemoteClusterForPDH seems to have a lot of memory-sharing between goroutines to accomplish "everyone gives up when one succeeds". The context library lets us do this sort of thing a way that separates the "choose first success" logic from the "do something that might succeed" logic.

```
success := make(chan resultType)
fails := make(chan error, len(blah))
ctx, cancel := context.WithCancel(parentCtx)
defer cancel()
var wg sync.WaitGroup
for range blah {
  wg.Add(1)
  go func() {
    defer wg.Done()
    result, err := dostuff(ctx)
    if err != nil {
      fails <- err
      return
    }
    select {
    case <-ctx.Done():
    case success <- result:
    }
  }()
}
go func() {
  wg.Wait()
```

```
  cancel()
}()

var result resultType
select {
case <-ctx.Done():
  // all failed, or parentCtx cancelled
  var errs []error
  for len(fails) > 0 {
    errs = append(errs, <-fails)
  }
  return fmt.Errorf("errors: %v", errs)
case result = <-success:
}
```

This way dostuff() can just do its stuff, and check ctx.Done()/Err() where convenient.

When copying the request ID into the error message text, it seems like we should use the one from the request currently being handled, not from the erroneous upstream response.

httpserver.GetRequestID(header) doesn't seem like an especially valuable feature, just a more verbose way to spell header.Get("X-Request-Id")...?

Should be consistent with http.StatusOK etc., instead of 200, 404, etc.

#13 - 10/30/2018 03:30 PM - Peter Amstutz

Tom Clegg wrote:

> In services/api/app/models/container.rb, the added "accept unreadable PDH as docker image" code seems like it belongs in models/collection.rb with the rest of the "choose PDH for docker image" code -- it seems like adding "if coll_match.empty? then return loc.to_s" there would avoid the duplicate code and database query.

Moved the logic over to find_all_for_docker_image.

> This change means a bad/unreadable docker image PDH isn't detected until a worker VM comes up and tries to load it, as the deleted test case indicates. Controller could catch this by using searchRemoteClusterForPDH before passing along the "create CR" req. I think it's OK to accept the regression for the time being on sites that actually use federation, but it seems like we might as well retain the previous Rails behavior on sites that don't have any remotes configured.

API server now checks remote_hosts. I did not (yet) add the check to arvados-controller because it's a little bit complicated -- the "search for PDH" code is part of a ServeHTTP() method, and needs to be refactored to be suitable to call as a standalone function as opposed to a request handler stack.

> Why rename (*proxy)Do (analogous to (*http.Client)Do()) to ForwardRequest?

Because I was splitting it into two functions, I wasn't sure if it made sense for one of those functions to keep the same name since it didn't have the same behavior. But since you brought it up, I renamed ForwardRequest() back to "Do()"

> In Do/ForwardRequest, why remove defer cancel()? Docs say we should call cancel.

Because you can't call cancel() until you've finished reading from the Body, but Do() no longer does response forwarding, instead the response is being returned to the caller. So I fixed it to return the cancel method (and it is now the caller's job to do "defer cancel()")

> The comments on ForwardResponse and ForwardRequest seem backwards: doesn't "downstream" normally mean "closer to client"?

Yea that's confusing. Fixed.

> In federation_test.go should the last two of the four new lines be swapped?

Yes. Fixed.

> The "shared mutex and pointers" approach in searchRemoteClusterForPDH seems to have a lot of memory-sharing between goroutines to accomplish "everyone gives up when one succeeds". The context library lets us do this sort of thing a way that separates the "choose first success" logic from the "do something that might succeed" logic.
>
> This way dostuff() can just do its stuff, and check ctx.Done()/Err() where convenient.

I rewrote it to use channels.

> When copying the request ID into the error message text, it seems like we should use the one from the request currently being handled, not from the erroneous upstream response.

This code is deleted, but noted for the future.

> httpserver.GetRequestID(header) doesn't seem like an especially valuable feature, just a more verbose way to spell header.Get("X-Request-Id")...?

Yes, I changed it into a constant. I just didn't want to repeat the "X-Request-Id" string literal.

> Should be consistent with http.StatusOK etc., instead of 200, 404, etc.

Done.

now 14262-remote-container @ b11eec52a05753cb587621393d5c4649940149fe

https://ci.curoverse.com/view/Developer/job/developer-run-tests/954/

#14 - 10/30/2018 06:17 PM - Peter Amstutz

Rebased, now 4427f2c5f740d03d5ee38745159f61b6805843e7

https://ci.curoverse.com/view/Developer/job/developer-run-tests/956/

#15 - 10/31/2018 03:08 PM - Peter Amstutz

- Target version changed from 2018-10-31 sprint to 2018-11-14 Sprint

#16 - 10/31/2018 05:03 PM - Peter Amstutz

Should fix tests

7f223f48c24dfa8c3d8247f8e48656a5edca7ea5

https://ci.curoverse.com/view/Developer/job/developer-run-tests/957/

#17 - 10/31/2018 05:25 PM - Tom Clegg

Aside: I don't want the merge to get stuck on this, but for the sake of getting better at things, I'll observe the proxy context handling still feels wrong -- I think we are simply creating the context/deadline in the wrong place. We should probably be creating the deadline in a place where we *could* write defer cancel() right away, and pass that context inwards, instead of creating the context later and passing the CancelFunc outwards.

In (h *collectionFederatedRequestHandler) ServeHTTP we're throwing away the returned cancel func and explaining why we don't need to obey the context rules. But it looks like we do in fact consume the response before the end of the func, so is there anything gained by not obeying the rules?

(*arvados.Client) MakeRequest

- doesn't seem to be used
- if needed, should be NewRequest just like http.NewRequest
- if needed, should have a doc comment
- doesn't need the check for an existing X-Request-Id header, or req.Header==nil, since it just created a new req itself and has already called req.Header.Set() without a panic
- seems to disagree with Do() about when/where Authorization and X-Request-Id headers should be added/overwritten

#18 - 11/01/2018 03:02 PM - Peter Amstutz

Tom Clegg wrote:

> Aside: I don't want the merge to get stuck on this, but for the sake of getting better at things, I'll observe the proxy context handling still feels wrong -- I think we are simply creating the context/deadline in the wrong place. We should probably be creating the deadline in a place where we *could* write defer cancel() right away, and pass that context inwards, instead of creating the context later and passing the CancelFunc outwards.

I moved it up to the top of the handler stack, so the deadline now applies to the entire request.

> In (h *collectionFederatedRequestHandler) ServeHTTP we're throwing away the returned cancel func and explaining why we don't need to obey the context rules. But it looks like we do in fact consume the response before the end of the func, so is there anything gained by not obeying the rules?

That code is gone (because the deadline moved and Do() no longer returns CancelFunc) but the reason it couldn't just defer cancel() was that at the point the function returns, we've only done a send on the success channel, we don't know if the consumer finished forwarding the response, and an

early cancel would close the body.

> (*arvados.Client) MakeRequest
>
> - doesn't seem to be used
> - if needed, should be NewRequest just like http.NewRequest
> - if needed, should have a doc comment
> - doesn't need the check for an existing X-Request-Id header, or req.Header==nil, since it just created a new req itself and has already called req.Header.Set() without a panic
> - seems to disagree with Do() about when/where Authorization and X-Request-Id headers should be added/overwritten

I don't want to die on this hill so I am reverting it. But:

- There's a couple of places that create their own request objects and then call DoAndDecode() and rely on it to set headers
- My intention was to use it for methods like genericFederatedRequestHandler.remoteQueryUUIDs() to build the request and then pass it to either localClusterRequest() or remoteClusterRequest(), but I had not done that part yet. So I'll save refactoring arvados.Client for when I refactor the other code.
- I'm frustrated that we have two redundant Go SDKs and now controller seems to be well on its way to becoming a third one.

#19 - 11/01/2018 03:05 PM - Peter Amstutz

6a7a7920e8ce4b6f6743d0a644afb87e6bae63c1

https://ci.curoverse.com/job/developer-run-tests/959/

#20 - 11/01/2018 06:32 PM - Tom Clegg

I think the correct way to fix the problem noted in 7f223f48c24dfa8c3d8247f8e48656a5edca7ea5 is to change source:services/api/test/integration/remote_user_test.rb so to remote_hosts = remote_hosts.merge('zbbbb' => @remote_host) instead of modifying the hash in place. Then the existing "restore original config" teardown hook should work, and other unrelated tests can go back to relying on a predictable config.

In fed_containers.go, currentUser.Authorization.UUID[0:5] == h.handler.Cluster.ClusterID panics if len(UUID)<5. Even though that "can't happen", I'd suggest strings.HasPrefix(currentUser.Authorization.UUID, h.handler.Cluster.ClusterID) for checking safely.

The same problem appears in some other recently-merged code too. On 4xphq, this produces a controller stack trace and an HTML-formatted 502 response from Nginx:

```
curl -H "Authorization: Bearer abcde" "https://$ARVADOS_API_HOST"'/arvados/v1/containers?filters=%5b%5b"uuid","="," "oops"%5d%5d'
```

The rest LGTM, thanks.

#21 - 11/01/2018 07:17 PM - Peter Amstutz

Tom Clegg wrote:

> I think the correct way to fix the problem noted in 7f223f48c24dfa8c3d8247f8e48656a5edca7ea5 is to change source:services/api/test/integration/remote_user_test.rb so to remote_hosts = remote_hosts.merge('zbbbb' => @remote_host) instead of modifying the hash in place. Then the existing "restore original config" teardown hook should work, and other unrelated tests can go back to relying on a predictable config.

Thanks for tracking down the root cause. Fixed.

> In fed_containers.go, currentUser.Authorization.UUID[0:5] == h.handler.Cluster.ClusterID panics if len(UUID)<5. Even though that "can't happen", I'd suggest strings.HasPrefix(currentUser.Authorization.UUID, h.handler.Cluster.ClusterID) for checking safely.

Fixed.

> The same problem appears in some other recently-merged code too. On 4xphq, this produces a controller stack trace and an HTML-formatted 502 response from Nginx:

Fixed.

> The rest LGTM, thanks.

11ed4d78b3abaa8f31e749093638df0804753ad4

https://ci.curoverse.com/view/Developer/job/developer-run-tests/961/

#22 - 11/05/2018 06:46 PM - Peter Amstutz

- Status changed from In Progress to Resolved

#23 - 11/13/2018 09:52 PM - Tom Morris

- Release set to 14