Pulp - Task #2090

Create a plan for user/auth in 3.0

07/20/2016 05:13 AM - mhrivnak

Status:	CLOSED - CURRENTRELEASE	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	ttereshc	**% Done:**	0%
Category:		**Estimated time:**	0:00 hour
Sprint/Milestone:			
Platform Release:		**Tags:**	Pulp 2
Groomed:	Yes	**Sprint:**	Sprint 9
Sprint Candidate:	Yes	**Quarter:**	

Description

This task is largely to determine if we should use django's built-in auth plus the auth in django-rest-framework. Hopefully yes, because using it gets us a lot for free, including easy integration with tools like django-admin.

This task is successful when reasonable answers to the below questions have been agreed on, and tasks have been created that cover at least the next steps, if not all of the work required.

Below are notes from the team on this subject:

```
Hopefully we can just replace what we have with django's auth.
Q: Will it meet user needs?
Q: If not, do we need to introduce a more detailed framework like Django Guardian? (https://github.com/lukaszb/django-guardian )
Q: Will we need to add custom authz logic? What's the scope?
Q: To what extent can/should we migrate existing authz data? Perhaps we should just not.
+1 to not doing this now, but we look at porting the data after we have the new one implemented. In other words, we could green-field use django's auth and decide later.
Q: What authn mechanisms will we support? ssl certs? cookies? basic? other? What work is involved for us to make these happen vs. what can django do out of the box?
Q: What will be our integration story with freeipa?
We could use generic Kerberos integration to work. Not only should it work in theory, but I believe rbarlow successfully tested a user patch for kerberos support against a freeipa based kerberos. One area then should be adding/testing Kerberos support from the start.
django-rest-framework (DRF) has pluggable auth, so DRF resources might be useful for implementing authnz features in the API:
http://www.django-rest-framework.org/api-guide/authentication/
http://www.django-rest-framework.org/api-guide/permissions/
http://www.django-rest-framework.org/topics/third-party-resources/#authentication
```

Related issues:

Related to Pulp - Task #946: Develop a plan to improve Pulp's authentication ...	**CLOSED - WONTFIX**
Related to Pulp - Task #2243: Create custom django User model	**CLOSED - CURRENTRELEASE**
Related to Pulp - Task #2356: Add serializer for the user model	**CLOSED - CURRENTRELEASE**
Related to Pulp - Task #2357: Add a user ViewSet	**CLOSED - CURRENTRELEASE**
Related to Pulp - Story #2358: As a user, I can authenticate with username an...	**CLOSED - CURRENTRELEASE**
Related to Pulp - Story #2359: As a user, I can use JWT tokens for authenticaton	**CLOSED - CURRENTRELEASE**
Related to Pulp - Task #2360: Create a plan for user authorization in 3.y	**CLOSED - WONTFIX**
Related to Pulp - Story #2361: As a user, I can authenticate using an externa...	**CLOSED - WONTFIX**
Related to Pulp - Story #2366: As a user, my password can be expired	**CLOSED - WONTFIX**
Related to Pulp - Story #2367: As a user, I can configure the expiration peri...	**CLOSED - DUPLICATE**
Has duplicate Pulp - Task #1874: Plan User/Auth system for 3.0	**CLOSED - DUPLICATE**

History

#1 - 07/20/2016 05:22 AM - mhrivnak

- Blocks Task #2091: Create a plan for the REST API in 3.0 added

#2 - 07/20/2016 03:17 PM - amacdona@redhat.com

While finishing user/auth does block finishing 3.0 Rest API, I don't think that this plan should block the planning for the rest API (2091)

#3 - 07/21/2016 03:01 AM - mhrivnak

We at least need to decide if and how we will use the auth features of django-rest-framework. I certainly agree that planning the REST API can begin before this task is done, but I don't think we can call #2091 complete until this one is complete.

#4 - 07/21/2016 03:10 AM - mhrivnak

That said, I'd rather have this groomed than worry about blocking relationships, so I'm ok removing it if people are otherwise happy with it. Any other feedback?

#5 - 07/21/2016 03:25 PM - ipanova@redhat.com

Can we just keep them related so we would at least keep in mind that these 2 piece of work are dependent?

#6 - 07/21/2016 03:36 PM - dkliban@redhat.com

- Groomed changed from No to Yes

#7 - 07/21/2016 04:24 PM - mhrivnak

- Sprint/Milestone set to 24

#8 - 07/29/2016 03:33 PM - bmbouter

FYI, maybe a helpful read for this topic

http://simpleisbetterthancomplex.com/tutorial/2016/07/22/how-to-extend-django-user-model.html

#9 - 08/02/2016 08:04 PM - ttereshc

This comment is just to express interest in working with someone on this or related to the user/auth issues.
For now I will try to figure out what we have/need and read/research on related topics.

#10 - 08/05/2016 05:19 PM - amacdona@redhat.com

It will not be possible to do a 1 to 1 migration because the Django ContentType system is inherently different than our "resource" based permissions system. I think it is probably worth going forward saying "we will not support migration of users/auth". This will give us the freedom to really do it right without having to worry about backwards compatibility of any kind. After implementing our new system, it might be worth it to create some helper scripts to ease the pain without making promises.

Some things that I think we probably could do:

- Migration of User objects should be trivial, but doing this without permissions is only limited value so it should not be run by default.
- Resources that point to a specific object. We might be able to use our Django url regexes to identify which objects a user has been given explicit permissions for, and convert them into the new style object permissions. Example: The resource '/repositories/zoo/' would translate to having permissions over the zoo repo object.
- We should also be able to add permissions for users with permissions over all objects. For example, if a user has permissions over the '/repositories/' resource, they would have access to all repository objects.

Perhaps our best option would be to encourage our users to generate their own scripts and share them, leaving us completely out of it.

#11 - 08/08/2016 04:09 PM - bmbouter

+1 to modeling the new user stuff without being concerned about porting data. As you mention, after we get the new user model done we can find ways to migrate the data later.

#12 - 08/11/2016 04:24 PM - mhrivnak

- Blocks deleted (Task #2091: Create a plan for the REST API in 3.0)

#13 - 08/15/2016 08:16 PM - mhrivnak

- Sprint/Milestone changed from 24 to 25

#14 - 08/18/2016 06:25 PM - ttereshc

Some thoughts:

User needs

I assume that user needs are defined by what we have now + what users use and ask for.
Currently the following options are available.
For authn:

- login/password in Pulp
- SSL certs (Pulp performs client cert validation against CA)
- various forms of authn configured in Apache (Basic auth, LDAP, Kerberos, ...)
  so that Pulp does not deal with authn

For authz:

- users and roles(=user groups) having access permissions to a particular resources (identified by URI)

Common authn use case:

- use some form of authn configured in Apache as a first step towards obtaining client-side SSL cert
  issued by Pulp subsequently used for authn performed jointly by Apache and Pulp

It is hard to estimate how heavily users/roles/permissions are used now.
Users' feedback ["[0]":https://www.redhat.com/archives/pulp-list/2016-July/msg00046.html\], ["[1]":
https://www.redhat.com/archives/pulp-list/2016-July/msg00047.html\] which includes requests for:

- ability to configure multiple forms of authn easily
- introduce access permissions for a group of the repositories so that users would be permitted
  to add/update/delete not all repositories but only inside this group

Hopefully we can just replace what we have with django's auth.

Q: Will it meet user needs?

Unless additional plugins are involved -- probably no.
For authn:

- login/password authn by Pulp/Django will work out of the box
- client side validation against Pulp CA may be moved to Apache(?) or may be there was a reason to do it that way
- various forms of authn:
  - could be the same as before - configuration in Apache
    - this approach gives flexibility to the user (most types of authn supported by Apache could be configured)
    - configuring several authn methods at a time may complicate Apache configuration
  - could alternatively be performed by Django using third-party or Pulp-custom authn backends
    - it is argueably harder to add new Django authn backend than to configure another Apache authn
    - multiple authn backends can be used
    - most of the authn methods are already implemented as plugins to Django in the form of custom authn backends

For authz:

- out of the box only model-level permissions can be set (e.g. access to Repository model)
- to grant access to a particular repo for example, object-level (instance-level) permissions should be used,
  there are several implementations available including django-guardian

Default Django User model has a number of limitations including length of the username which is 30 characters at most.
We do not know the max length of the username currently used and for various forms of authn 30 chars may not be enough,
if e-mail is used as a username, for example.

So custom User model will likely be needed.

Q: If not, do we need to introduce a more detailed framework like Django Guardian? (https://github.com/lukaszb/django-guardian)

I think so, because model-level permissions are not enough, we need object-level ones as well.

Q: Will we need to add custom authz logic? What's the scope?

In most cases features form all the auth plugins which we are going to use (like django-guardian) will be enough.
I have some concerns in regards to how manageable those users/groups/permissions are and how easy would be to accidentally assign inconsistent permissions.
One can grant conflicting permissions to the user and their expectations may differ from the way Django handles it.
Maybe just a detailed documentation on this topic will be sufficient.

Another potential customization to think about: how to grant permissions for a group of objects (asked by Kodiak) with possibility to add/modify/remove objects from this group.

Q: What authn mechanisms will we support? ssl certs? cookies? basic? other?

What work is involved for us to make these happen vs. what can django do out of the box?

It would be good to provide users with various number of authn mechanisms.
It looks like SSL certs are used by our users.
Client session ID or signed token (stored as cookie) could be a more lightweight way than SSL certs to aim the same goal - not to require login/password auth on each request(?).
Theoretically, cookies are vulnerable to stealing (AKA man in the middle attack) while SSL certs are not.
However, the way Pulp currently operates renders SSL security to cookie strength -- problem is that client private key is being generated by Pulp and sent down to client (along with the cert), afaik.
Some details on session ID vs tokens.

Django has two auth backends, one is for Django auth and the other is for external auth with REMOTE_USER being set by Apache in our case.
Other auth backends can be implemented and added if needed.
Most likely there is already an implementation of one or the other authn mechanism which we may want to add, so hopefully there is not so much work is involved for us to add another auth mechanism.

Any ideas on what authn mechanisms are mostly used? Just Basic auth? LDAP+Basic auth? LDAP+Kerberos?
What authn mechanisms we have to support and what are the nice-to-have ones?

I would also appreciate someone telling me a little bit more about content authentication mechanisms in Pulp. I am not sure I fully understand them yet. ;)

#15 - 08/26/2016 08:33 PM - mhrivnak

This is a great start. Thanks @tteresch ! I'll add some thoughts and reactions in no particular order:

For the user model, it's a shame that the username field is likely a deal-breaker. We probably want to just use a TextField without a size limit. Are we ready to make a task to create a new user model in django? Here are docs about how to do it:

https://docs.djangoproject.com/en/1.8/topics/auth/customizing/#substituting-a-custom-user-model

and how to make it compatible with django-admin:

https://docs.djangoproject.com/en/1.8/topics/auth/customizing/#custom-users-and-django-contrib-admin

For authz, we need some research, but it seems like one of the lower priority items for this overall task. Creating users and logging in are the first two things that need to be implemented.

For authn, client SSL certs are awesome, but not very common for web apps and services. I'd be very happy seeing pulp move to something more widely used for general web app authn. What do our peer projects do? Katello, foreman, candlepin, RHEV, ManageIQ, Openshift, etc. have all had to solve this problem. Hopefully we can make cross-project integration easier by participating in the same authn ecosystem as them. I think the next step here is to query some of those projects, and maybe also query the freeipa folks for their best practice suggestions.

Content auth is completely separate. This is the way in which we require a client such as yum to present an entitlement certificate when trying to access yum repositories. The way in which this authn and authz happens will continue to be driven by candlepin, and we are just along for the ride as enforcers.

#16 - 09/01/2016 03:01 PM - amacdona@redhat.com

- Related to Task #946: Develop a plan to improve Pulp's authentication offerings added

#17 - 09/02/2016 04:45 PM - rglaue

Please note, SessionIds in cookies are not RESTfully compliant, because the client's authorization is stored on the server.
See Also: http://stackoverflow.com/questions/6068113/do-sessions-really-violate-restfulness

We want to keep Basic Auth - we use it alot for one-off tasks. However, we'd love to have something more REST friendly, like JSON Web Tokens (JWT), that we can use in place of Basic Auth.
https://jwt.io/
JWT is RESTfully compliant. It is stateless because "...the token is a self-contanined entity that conveys all the user information."
See Also: http://blog.brainattica.com/restful-json-api-jwt-go/

We can have the JWT support from the django-rest-framework-jwt project for the Django REST framework.
https://github.com/GetBlimp/django-rest-framework-jwt

For us, the REST API is very important. We use it to integrate Pulp with Jenkins for Continuous Integration.
We currently use cert-based auth for connections from our continuous integration environment, managed by Jenkins.
JWT is also usable.

Also, with REST, we can build a RESTful WebUI on top: https://github.com/marmelab/ng-admin.git (Basic Auth and JWT are supported)

#18 - 09/02/2016 06:18 PM - bmbouter

- Related to Story #227: [RFE] Allow wildcards in permissions added

#19 - 09/02/2016 06:20 PM - bmbouter

- Has duplicate Task #1874: Plan User/Auth system for 3.0 added

#20 - 09/06/2016 12:41 AM - mhrivnak

- Sprint/Milestone changed from 25 to 26

#21 - 09/08/2016 06:27 PM - ttereshc

- Status changed from NEW to ASSIGNED

- Assignee set to ttereshc

#22 - 09/12/2016 09:43 PM - ttereshc

- Related to Task #2243: Create custom django User model added

#25 - 10/03/2016 03:36 PM - mhrivnak

- Sprint/Milestone changed from 26 to 27

#26 - 10/19/2016 04:29 PM - ttereshc

authentication proposal https://www.redhat.com/archives/pulp-dev/2016-September/msg00049.html

#27 - 10/19/2016 06:57 PM - ttereshc

- Related to Task #2356: Add serializer for the user model added

#28 - 10/19/2016 06:58 PM - ttereshc

- Related to Task #2357: Add a user ViewSet added

#29 - 10/19/2016 06:58 PM - ttereshc

- Related to Story #2358: As a user, I can authenticate with username and password stored in Pulp added

#30 - 10/19/2016 06:58 PM - ttereshc

- Related to Story #2359: As a user, I can use JWT tokens for authenticaton added

#31 - 10/19/2016 07:12 PM - ttereshc

- Related to Task #2360: Create a plan for user authorization in 3.y added

#32 - 10/24/2016 03:01 AM - ttereshc

- Related to Story #2361: As a user, I can authenticate using an external authority added

#33 - 10/24/2016 03:02 AM - ttereshc

- Related to Story #2366: As a user, my password can be expired added

#34 - 10/24/2016 03:02 AM - ttereshc

- Related to Story #2367: As a user, I can configure the expiration period for JWT tokens added

#35 - 10/24/2016 03:02 AM - ttereshc

- Checklist item [x] Answer the questions in the notes and get buy-in from the team. set to Done

#36 - 10/24/2016 03:02 AM - ttereshc

- Checklist item [x] Create tasks covering the next steps for this work, and optionally as much of the work as can be reasonably defined. set to Done

#37 - 10/24/2016 03:11 AM - ttereshc

- Status changed from ASSIGNED to CLOSED - CURRENTRELEASE

- Duplicating here the authentication proposal as a result of this task: https://www.redhat.com/archives/pulp-dev/2016-September/msg00049.html
- Task #2360 for tracking authorization plans created.
- All the stories for implementing user authentication created and linked to this task.

#38 - 10/24/2016 03:12 AM - ttereshc

- Related to deleted (Story #227: [RFE] Allow wildcards in permissions)

#39 - 03/08/2018 08:10 PM - bmbouter

- Sprint set to Sprint 9

#40 - 03/08/2018 08:11 PM - bmbouter

- Sprint/Milestone deleted (27)

#41 - 04/15/2019 10:27 PM - bmbouter

- Tags Pulp 2 added