django-mellon - Development #32376

Introduire une distinction entre rôles statiques et dynamiques

16 avril 2019 14:03 - Valentin Deniaud

Statut:	Fermé	**Début:**	16 avril 2019
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Fonctionne en miroir avec #33708.

Dans la norme RBAC, on peut lire :

> A subject may have multiple simultaneous sessions with/in different roles.

C'est la philosophie de ce ticket : on a des rôles statiques, affectés à l'usager en base de donnée, et des rôles dynamiques liés à la session courante.
La distinction est introduite à la connexion de l'usager, l'IdP envoie deux listes de rôles, statiques et dynamiques. À l'application de limiter les permissions de l'usager à ses rôles dynamiques, et de demander les rôles seulement statiques à l'IdP si besoin.

Dans le cadre de l'authent multi-facteurs, l'IdP génère la liste des rôles dynamiques en fonction du niveau d'authentification de l'usager, que lui seul connaît et maintient.

Demandes liées:

Lié à Authentic 2 - Development #32786: Authentification multi-facteurs	**Fermé**	**03 mai 2019**
Lié à Authentic 2 - Development #33708: Multi-facteurs : interactions avec le...	**Fermé**	**05 juin 2019**

Historique

#1 - 16 avril 2019 14:07 - Valentin Deniaud

- Fichier 0001-models-add-authentication-level-attribute-to-Group.patch ajouté

- Fichier 0002-views-handle-authentication-levels-requests-and-resp.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

#2 - 16 avril 2019 15:11 - Benjamin Dauvergne

Je ne vois pas de lien entre le premier et le deuxième patch; ensuite on ne veut pas forcer le format des classes d'authentification, au mieux on aura un dico qui mappe les URLs vers des niveaux :

```
MAPPING = {
  'http://www.entrouvert.com/auth-level/1': 1,
  'http://www.entrouvert.com/auth-level/2': 2,
  'http://www.entrouvert.com/auth-level/3': 3,
}
```

Ceci pour pouvoir accepter d'autres niveaux d'authentification, pour l'application au niveau applicatif j'ai de toute façon l'idée que le passage par les groupes/rôles est une voie sans issue, il sera certainement plus simple d'avoir un middleware forçant un niveau par rapport à des chemins ({'/manage/': {'min_auth_level': 2}).

Actuellement nos applications ne savent pas vraiment qu'elles fonctionnent avec django-mellon et du SAML, on met un décorateur @login_required sur une vue et c'est terminé, ce serait bien de rester proche de cette simplicité.

#3 - 16 avril 2019 15:55 - Valentin Deniaud

Super idée le mapping, ça balaye une couche de saleté déjà !

Sinon ce ticket est relié à d'autres et vise à montrer comment tout marche ensemble, vu qu'on en est pas encore à merge des trucs. Du coup le premier patch est utilisé par hobo pour l'ajout des niveaux (#32381) et par chrono (brique que j'ai choisie arbitrairement comme exemple) pour les lire et faire ses vérifs (#32379).

> il sera certainement plus simple d'avoir un middleware forçant un niveau par rapport à des chemins ({'/manage/': {'min_auth_level': 2})

Plus simple, ça c'est clair ! Mais avoir une condition de ce genre ça enlève pas mal l'intérêt de configurer les choses au niveau rôles (ça ressemblerait plutôt à un flag « activer l'authentification multifacteur sur cette brique »).
Après il me semble avoir déjà soulevé la question de l'utilité d'avoir un système aussi granulaire, et que la conclusion c'était « au pire j'essaye et on voit ce que ça donne » (on est à l'étape où on voit ce que ça donne).

> nos applications ne savent pas vraiment qu'elles fonctionnent avec django-mellon

J'ai essayé de rester dans cet esprit là.

> on met un décorateur @login_required sur une vue et c'est terminé

Hum, j'ai pas regardé toutes les briques mais dans chrono c'est loin d'être le cas, à chaque vues derrière /manage/ on regarde dans quels groupes est l'utilisateur et on prend des décisions sur quoi afficher, et c'est à ce niveau là que j'opère.

> ce serait bien de rester proche de cette simplicité

J'ai essayé aussi, et je suis resté dans l'esprit décorateur pour cacher la complexité supplémentaire et n'avoir à se préoccuper de rien quand on code des vues.

#4 - 16 avril 2019 17:55 - Valentin Deniaud

- Fichier 0001-use-a-mapping-instead-of-hard-coded-string.patch ajouté

#5 - 16 avril 2019 23:28 - Benjamin Dauvergne

Comme tu t'es lancé pas mal dans les interactions entre rôle et contrôle d'accès je pense que ce serait important que tu te familiarises avec la norme ANSI RBAC[1] qui a donné l'implémentation actuelle de RBAC dans authentic, et notamment le concept de session; il y a clairement une notion statique (l'affectation des utilisateurs aux rôles) et dynamique (la mise en pratique du rôle à un moment donné) qui devrait je pense simplifier ta réflexion sur la propagation des niveaux d'authentification.

[1] https://profsandhu.com/journals/tissec/ANSI+INCITS+359-2004.pdf

Il me semble qu'on pourrait imaginer une implémentation proche : l'affectation des utilisateurs aux rôles (ou aux groupes au niveau des applications) serait la vision statique des rôles, les rôles obtenus au SSO la vision dynamique, celle-ci pouvant dépendre des niveaux d'authentification. Les décisions au cours d'une session ne doivent être prise qu'en fonction des rôles dynamiques, les décisions hors session (ex.: envoyer un mail aux membres d'un groupe) se font en fonction de la vision statique des rôles.

Il resterait le problème d'un changement de l'association statique alors qu'une session est en cours (provisionning pendant qu'un utilisateur est loggé), dans ce cas je proposerai de stocker en session l'association statique au moment de la définition de l'association dynamique est de la comparer à l'association statique réelle au moment de l'évaluation de l'association dynamique : si un rôle n'est plus là statiquement, il ne saurait plus l'être dynamiquement, si un nouveau rôle statique est apparu, il faut interroger l'IdP dans la session pour vérifier s'il faut aussi l'ajouter à l'association dynamique pour cette session.

On n'a actuellement pas de conceptualisation des sessions distribuée entre l'IdP et les applications (ça donne quand même lieu à des identifiants à plusieurs endroits, le cookie de session sur l'IdP, le sessionIndex en SAML et les access token distribuées en OIDC au cours d'une même session, ces 3 aspects sont liés entre eux).

L'approche me semble plus saine parce qu'elle ne déporte pas la prise décision dynamique dans les applications; s'il le faut l'IdP serait réinterrogé.

#6 - 17 avril 2019 18:23 - Valentin Deniaud

C'est bien intéressant tout ça, merci.

> L'approche me semble plus saine parce qu'elle ne déporte pas la prise décision dynamique dans les applications;

En préambule je souligne quand même que dans l'approche que j'ai implémenté, il y a effectivement un ajout de prise de décision dynamique, mais qu'il se superpose à une prise de décision déjà présente. À la condition en place « l'utilisateur a-t-il ce rôle » vient s'ajouter « si oui, l'utilisateur a-t-il un niveau d'authentification suffisant pour en faire usage ». C'est à distinguer d'une solution qui viendrait introduire de la logique à de nouveaux endroits. Mais je ressens effectivement le côté « tant qu'y en a un ça va, c'est quand il y en a beaucoup que ça pose un problème », dû au fait de transformer du code trivial et critique, deux propriétés qu'on aime bien voir ensemble, en code plus complexe. Et aussi que cette logique soit destinée à être copiée collée dans toutes les applications.
Je suis d'accord que ça serait idéal de ne rien ajouter !

> s'il le faut l'IdP serait réinterrogé.

C'est là tout le problème. Si la logique « si l'utilisateur n'a pas le bon rôle, interroger l'idp », qui est une forme de prise de décision, certes allégée, vient s'ajouter dans les applications en lieu est place de la vérification du niveau, ça rend quand même l'approche un poil caduque. Surtout que cette logique en appelle d'autres, genre « se rappeler qu'on a déjà demandé et qu'il y a eu un refus », par ex.

Donc il faudrait viser pour de vrai le 0 ajout dans les applications, et j'ai du mal à imaginer comment la jouer niveau code.
Pour moi ça voudrait dire tout faire dans django-mellon, mais sans ajouter des champs ou des nouvelles méthodes publiques aux modèles User ou Group pour garder le côté « appli réutilisable ». Ça donnerait par exemple un modèle User proxy, avec un User.groups qui va chercher dans Group (statique) si l'utilisateur n'est pas connecté, et dans les groupes dans la session (dynamique) dans le cas inverse.

Ensuite, prenons la fonction toute simple qui gère l'essentiel des permissions dans chrono :

```
def can_be_managed(self, user):
    group_ids = [x.id for x in user.groups.all()]
    return bool(self.edit_role_id in group_ids)
```

Il faudrait à minima réécrire comme suit :

```
def can_be_managed(self, user):
    return user.groups.filter(id=self.edit_role_id).exists()
```

Et là on peut peut-être s'en sortir en modifiant le Manager, en faisant en sorte de checker d'abord les groupes dans la session (qui n'est d'ailleurs pas accessible depuis l'objet User, mais on doit pouvoir ruser), puis que si le groupe n'existe pas dedans il faut aller voir les groupes statiques (comportement normal du Manager), et que si il y est il faut lever une exception style PermissionDenied qui demande la montée de niveau.
Je sais pas dans quelle mesure c'est faisable, et on a un petit soucis de « explicit is better than implicit ». En effet, on ne s'attend vraiment pas à ce qu'un queryset puisse lever des exceptions custom... L'alternative serait de définir des nouvelles méthodes, et d'avoir des checks if 'mellon'... pour les appeler ou non.

Last but not least, il va y avoir des moments où on va vouloir taper dans les rôles statiques d'un utilisateur connecté, par exemple pour afficher un bouton (et le clic sur ce bouton provoquera ensuite la montée) ou un message (« cliquez ici pour obtenir un rôle supérieur et voir le contenu de cette cellule »). Là il va falloir établir une distinction en changeant le code de la brique, à base de can_be_managed(user, check_static=True) ou d'une nouvelle méthode, possiblement dans mellon.

Voilà je veux bien ton avis, notamment si tu penses que les pistes évoquées ici vont pas dans la bonne direction, sinon je me mets à bricoler et à voir comment les choses se goupillent.

#7 - 18 avril 2019 18:08 - Valentin Deniaud

- Statut changé de Solution proposée à En cours

Je pousse sûrement la PoC de cette nouvelle approche demain, c'est pas si compliqué.

#8 - 23 avril 2019 11:45 - Valentin Deniaud

- Fichier 0001-views-handle-authentication-level-increase-requests.patch ajouté

- Fichier 0002-views-save-role-slug-SAML-values-in-session.patch ajouté

- Fichier 0003-utils-add-check_session_roles-decorator.patch ajouté

- Statut changé de En cours à Solution proposée

#9 - 23 avril 2019 12:01 - Valentin Deniaud

Voilà, j'ai implémenté l'histoire de rôles statiques/dynamiques. Difficile de comparer avec l'approche précédente vu que j'ai eu de nouvelles idées à droite à gauche pour simplifier le tout, mais en gros ça revient à remplacer le test du niveau d'authentification par le test du rôle statique ou dynamique.

Un side-effect sympa c'est que ça ne sert plus à rien d'avoir des nouvelles URI pour représenter les niveaux d'authentification.
Plus précisément on observe que mellon n'a plus besoin de regarder à quel niveau correspond 'http://www.entrouvert.com/auth-level/1'. En revanche au moment de demander des rôles en plus il faut qu'il sache à quelles URI correspond le niveau x.
Donc il semble qu'on puisse très bien mettre en place un mapping du type :

```
MAPPING = {
    2: ('urn:oasis:names:tc:SAML:2.0:ac:classes:X509', 'urn:oasis:names:tc:SAML:2.0:ac:classes:MobileTwoFactor
Contract'),
    3: ('urn:oasis:names:tc:SAML:2.0:ac:classes:PGP',),
}
```

À configurer à la volée quelque part dans a2, et à transmettre aux autres briques via hobo.

Mais avant ça, un avis sur cette nouvelle version ?

#10 - 24 avril 2019 11:46 - Valentin Deniaud

- Fichier 0001-adapters-add-a-separate-method-to-remove-superuser-f.patch ajouté

- Fichier 0002-views-save-is_staff-in-session.patch ajouté

- Fichier 0003-utils-add-is_staff-check-to-user_has_role.patch ajouté

#11 - 24 avril 2019 11:49 - Valentin Deniaud

Mis à jour pour gérer le cas du is_staff. Clairement plus simple à faire avec la nouvelle approche :)

#12 - 24 avril 2019 14:31 - Valentin Deniaud

> Il resterait le problème d'un changement de l'association statique alors qu'une session est en cours

J'allais me lancer là dedans, et puis en fait ça marche déjà, sans besoin de stocker les rôles statiques dans la session. Cela grâce à la logique de montée de niveau d'authentification : regarder d'abord dans les rôles statiques de l'utilisateur, et si le rôle est présent, regarder dans la session. Si il y est, on est bon, sinon demander plus plus à l'IdP.
Concrètement, si un rôle a disparu pour cause de provisionning, on n'ira même pas regarder dans la session et on dira non, si un rôle est apparu, on ne le trouvera pas dans la session et on réinterrogera.

#13 - 24 avril 2019 14:32 - Valentin Deniaud

- Fichier 0001-utils-add-is_staff-check-to-user_has_role.patch ajouté

#14 - 03 mai 2019 18:29 - Valentin Deniaud

- Lié à Development #32786: Authentification multi-facteurs ajouté

#15 - 16 mai 2019 14:01 - Benjamin Dauvergne

Il faudrait virer tout ce qui concerne les niveaux d'authentification, et passer comme AuthnClassRef une liste d'URL de la form :

https://entrouvert.com/authn-class-ref/role-uuid/<uuid-du-role>/

Ça évite d'inventer un truc au niveau de SAML (on réutilise AuthnClassRef) coté A2 on interprète ça comme le souhait d'avoir ces rôles, et on voit comment réagir.

#16 - 05 juin 2019 14:49 - Valentin Deniaud

- Fichier 0004-utils-helper-method-to-check-if-a-user-has-a-role.patch ajouté

- Fichier 0002-adapters-add-a-separate-method-to-remove-superuser-f.patch ajouté

- Fichier 0005-utils-add-check_session_roles-decorator.patch ajouté

- Fichier 0001-views-handle-role-requests.patch ajouté

- Fichier 0003-views-save-is_staff-in-session.patch ajouté

#17 - 05 juin 2019 14:51 - Valentin Deniaud

Benjamin Dauvergne a écrit :

> Il faudrait virer tout ce qui concerne les niveaux d'authentification, et passer comme AuthnClassRef une liste d'URL de la form :
>
> https://entrouvert.com/authn-class-ref/role-uuid/<uuid-du-role>/
>
> Ça évite d'inventer un truc au niveau de SAML (on réutilise AuthnClassRef) coté A2 on interprète ça comme le souhait d'avoir ces rôles, et on voit comment réagir.

Hop, fait.

#18 - 05 juin 2019 17:51 - Valentin Deniaud

- Lié à Development #33708: Multi-facteurs : interactions avec les SP ajouté

#19 - 17 juillet 2019 17:06 - Valentin Deniaud

- Fichier 0004-utils-add-method-to-check-if-a-user-has-a-role.patch ajouté

- Sujet changé de Support des niveaux d'authentification à Introduire une distinction entre rôles statiques et dynamiques

- Description mis à jour

Viré un décorateur au profit d'un middleware, et changé une méthode pour qu'elle accepte une liste de rôles plutôt qu'un seul. Plus de patch 5, tout dans 4.

#20 - 18 juin 2020 10:04 - Valentin Deniaud

- Assigné à Valentin Deniaud supprimé

#21 - 22 novembre 2021 11:25 - Valentin Deniaud

- Statut changé de Solution proposée à Fermé

Fichiers

0001-models-add-authentication-level-attribute-to-Group.patch	2,15 ko	16 avril 2019	Valentin Deniaud
0002-views-handle-authentication-levels-requests-and-resp.patch	3,63 ko	16 avril 2019	Valentin Deniaud
0001-use-a-mapping-instead-of-hard-coded-string.patch	3,83 ko	16 avril 2019	Valentin Deniaud
0001-views-handle-authentication-level-increase-requests.patch	5 ko	23 avril 2019	Valentin Deniaud
0002-views-save-role-slug-SAML-values-in-session.patch	913 octets	23 avril 2019	Valentin Deniaud
0003-utils-add-check_session_roles-decorator.patch	1,46 ko	23 avril 2019	Valentin Deniaud
0001-adapters-add-a-separate-method-to-remove-superuser-f.patch	1,95 ko	24 avril 2019	Valentin Deniaud
0002-views-save-is_staff-in-session.patch	3,63 ko	24 avril 2019	Valentin Deniaud
0003-utils-add-is_staff-check-to-user_has_role.patch	874 octets	24 avril 2019	Valentin Deniaud
0001-utils-add-is_staff-check-to-user_has_role.patch	896 octets	24 avril 2019	Valentin Deniaud
0004-utils-helper-method-to-check-if-a-user-has-a-role.patch	2,3 ko	05 juin 2019	Valentin Deniaud
0002-adapters-add-a-separate-method-to-remove-superuser-f.patch	1,95 ko	05 juin 2019	Valentin Deniaud
0005-utils-add-check_session_roles-decorator.patch	1,89 ko	05 juin 2019	Valentin Deniaud
0001-views-handle-role-requests.patch	2,16 ko	05 juin 2019	Valentin Deniaud
0003-views-save-is_staff-in-session.patch	3,59 ko	05 juin 2019	Valentin Deniaud
0004-utils-add-method-to-check-if-a-user-has-a-role.patch	3,76 ko	17 juillet 2019	Valentin Deniaud