Authentic 2 - Development #35482

passer le HASH du mot de passe des nouveaux utilisateurs aux service web d'A2

20 août 2019 15:28 - Nicolas Roche

Statut:	Fermé	**Début:**	20 août 2019
Priorité:	Normal	**Echéance:**	
Assigné à:	Nicolas Roche	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Pour signalepublik, il se pose la question de comment gérer le mot de passe de l'utilisateur qui déploie et devra accèder à une nouvelle instance.

Actuellement il est envisagé que l'utilisateur nous fournisse sont mot de passe via un formulaire WCS, puis que nous le passions à A2 lors de la création du compte par appel webservice. Mais entre temps le mot de passe est stocké en clair dans un fichier.

Pour faire un peu plus propre, il faudrait que ce soit le hash du mot de passe qui soit transmis. Donc, il faudrait alors pouvoir transformer le hash fourni par WCS (md5 ou sha1) ...

```
form_password_cleartext: secret
form_password_md5: 5ebe2294ecd0e0f08eab7690d2a6ee69
form_password_sha1: e5e9fa1ba31ecd1ae84f75caaa474f3a663f05f4
```

... en l'attribut password d'un objet User qui est une chaîne plus ou moins dans ce format :

```
<algorithm>$<iterations>$<salt>$<hash>
```

ex:

> pbkdf2_sha256$36000$DB8SFk2u19Kpbkdf2_sha256$36000$DB8SFk2u19KQ$H4ANoXyiYuy0auzZ8UFcLsC6iu0uvPhn89eMxe4aq98=Q$H4ANoXyiYuy0auzZ8UFcLsC6iu0uvPhn89eMxe4aq98=

Cela fonctionne ...

```
$ authentic-multitenant-manage shell
In : from authentic2.hashers import *
In : hasher = SHA1OLDAPPasswordHasher()
In : hasher.encode('secret', '')
Out: u'sha-oldap$$e5e9fa1ba31ecd1ae84f75caaa474f3a663f05f4'

$ psql
> \c authentic_multitenant
> set search_path to authentic_dev_publik_love;
> update custom_user_user set password = 'sha-oldap$$e5e9fa1ba31ecd1ae84f75caaa474f3a663f05f4' where email = 'nroche@localhost';
```

Je pensais dégrader la robustesse du mot de passe stocké,

> Par défaut, Django utilise l'algorithme PBKDF2 avec une fonction de hachage SHA256, un mécanisme d'étirement de mot de passe recommandé par le NIST. Cela devrait suffire pour la plupart des utilisateurs : c'est un algorithme bien sécurisé et exigeant d'énormes quantités de puissance de calcul pour être cassé.

... surtout avec la perte du salage (qui est ajouté au mot de passe avant le hashage). Mais en fait Authentic met à jour l'empreinte lors de la connection.

```
> select email, password from custom_user_user;
nroche@localhost | pbkdf2_sha256$36000$ujKn9GGKd4EX$L0Rm5F5qFsbn3Jp6FM06GUVJfBRwEHRFhv+4xZZCDEM=
```

D'où ce ticket pour voir si l'on pourrait étendre l'API webservice user pour prendre éventuellement un hash à la place du mot de passe.

Demandes liées:

Lié à w.c.s. - Development #35533: stockage des mots de passe hashés	**Rejeté**	**22 août 2019**

Révisions associées

Révision c98f24d1 - 05 septembre 2019 15:31 - Nicolas Roche

api: add a hashed_password attribute for user api (#35482)

Historique

#1 - 21 août 2019 22:51 - Benjamin Dauvergne

Oui c'est une bonne approche, à toi de voir comment tu récupères le hash depuis les appels au WS (nouveau champ, nouveau type de valeur pour le champ existant "password").

#2 - 22 août 2019 09:42 - Nicolas Roche

A priori ce serait un nouveau champ.

Par contre Thomas n'est pas très chaud parce qu'il n'y a pas d'API Django de prévue à cet effet et plus particulièrement à cause du Save qui utilise un attribut privé.
django/contrib/auth/base_user.py::AbstractBaseUser :

```
    # Stores the raw password if set_password() is called so that it can
    # be passed to password_changed() after the model is saved.
    _password = None
...
    def save(self, *args, **kwargs):
        super().save(*args, **kwargs)
        if self._password is not None:
            password_validation.password_changed(self._password, self)
            self._password = None
```

Perso je dirais que l'attribut _password est remis à None après utilisation et donc que "ça passe".
Tu en penses quoi ?

#3 - 22 août 2019 10:41 - Benjamin Dauvergne

Nicolas Roche a écrit :

> A priori ce serait un nouveau champ.
>
> Par contre Thomas n'est pas très chaud parce qu'il n'y a pas d'API Django de prévue à cet effet et plus particulièrement à cause du Save qui utilise un attribut privé.
> *django/contrib/auth/base_user.py::AbstractBaseUser* :
> [...]
>
> Perso je dirais que l'attribut _password est remis à None après utilisation et donc que "ça passe".
> Tu en penses quoi ?

Je ne connaissais pas, ça doit être récent (django 1.11) je dirai qu'on s'en fout, c'est juste pour valider le format du mot de passe, et c'est l'implémentation Django de ce truc, on en a une autre dans a2 qu'on applique pas non plus sur les WS il me semble.

Bon après pour reprendre le cas d'usage évoqué de base je ne vois pas ce que ça nous apporte, ok le mot de passe n'est pas envoyé en clair à a2 mais il est de toute façon conservé en clair dans la demande coté w.c.s. vu l'implémentation de PasswordField. Je rate un truc ?

#4 - 22 août 2019 10:55 - Thomas Noël

- Fichier Capture d'écran de 2019-08-22 10-52-38.png ajouté

Benjamin Dauvergne a écrit :

> Bon après pour reprendre le cas d'usage évoqué de base je ne vois pas ce que ça nous apporte, ok le mot de passe n'est pas envoyé en clair à a2 mais il est de toute façon conservé en clair dans la demande coté w.c.s. vu l'implémentation de PasswordField. Je rate un truc ?

Oui, wcs peut ne rien conserver en clair et ne garder que le hash. En revanche ça va pas plus loin que md5 et sha1 pour l'instant (en évolution, on pourrait ici profiter de la présence de Django pour ajouter les hashers de Django allez hop #35533.

#5 - 22 août 2019 11:06 - Benjamin Dauvergne

Ok j'ai lu que le code de PasswordEntryWidget, pas vu que PasswordField limiter les formats, ça a un sens alors. Pour le format du hash de toute façon Django mets à jour vers le format souhaité à chaque login si nécessaire, ce n'est donc pas bien grave.

#6 - 04 septembre 2019 18:33 - Nicolas Roche

- Fichier 0001-api-add-a-hashed_password-attribute-for-user-api-354.patch ajouté

- Statut changé de Nouveau à En cours

- Patch proposed changé de Non à Oui

Vu #35533, la construction des hash se fera en amont dans wcs.
Je n'ai pas trouvé comment valider (proprement) le format du hash passé.

La doc django (https://docs.djangoproject.com/fr/1.11/topics/auth/passwords/) indique :

```
<algorithm>$<iterations>$<salt>$<hash>
```

ce qui exclus par exemple les anciens formats sans itérations gérés par A2 :

```
<algorithm>$<salt>$<hash>
```

Est-ce que d'après vous je peux me baser sur ces 2 formats ou est-ce
que j'accepte sans regarder ?

#7 - 04 septembre 2019 23:13 - Benjamin Dauvergne

Nicolas Roche a écrit :

> Vu #35533, la construction des hash se fera en amont dans wcs.
> Je n'ai pas trouvé comment valider (proprement) le format du hash passé.

Tu as une fonction identify_hasher dans django.contrib.auth.hashers pour faire ça, ce n'est pas une API très publique mais elle est là.

> La doc django (https://docs.djangoproject.com/fr/1.11/topics/auth/passwords/) indique :
> [...]
> ce qui exclus par exemple les anciens formats sans itérations gérés par A2 :
> [...]
>
> Est-ce que d'après vous je peux me baser sur ces 2 formats ou est-ce
> que j'accepte sans regarder ?

Tu peux tenter un hasher.safe_summary(encoded) et refuser ce qui est envoyé si ça retourne la moindre exception, mais clairement sans le mot de passe en clair et une petite connaissance de l'algo en question t'as aucun moyen de savoir si ce qu'on t'envoie à un sens ou si ça peut être dangereux; genre on peut t'envoyer un nombre d'itération monstrueux ce qui est en fait un DOS (la prochaine fois que cette personne va se connecter ça va bloquer un worker jusqu'à ce que ça timeout, parce que ça va faire tenter de faire des milliards de calculs de hash).

#8 - 05 septembre 2019 14:42 - Nicolas Roche

- Fichier 0001-api-add-a-hashed_password-attribute-for-user-api-354.patch ajouté

- Statut changé de En cours à Solution proposée

> Tu peux tenter un hasher.safe_summary(encoded) et refuser ce qui est envoyé si ça retourne la moindre exception,

merci, du coup j'ai fait ça.

#9 - 05 septembre 2019 15:19 - Benjamin Dauvergne

- Statut changé de Solution proposée à Solution validée

Wunderbar.

#10 - 05 septembre 2019 15:33 - Nicolas Roche

- Statut changé de Solution validée à Résolu (à déployer)

```
commit c98f24d13c4d790c0eb61c20afcfa297a4901edb
Author: Nicolas ROCHE <nroche@entrouvert.com>
Date:   Wed Sep 4 18:02:32 2019 +0200

    api: add a hashed_password attribute for user api (#35482)
```

#11 - 09 septembre 2019 13:16 - Nicolas Roche

- Lié à Development #35533: stockage des mots de passe hashés ajouté

#12 - 11 septembre 2019 16:15 - Frédéric Péters

- Statut changé de Résolu (à déployer) à Solution déployée

Fichiers

Capture d'écran de 2019-08-22 10-52-38.png	4,69 ko	22 août 2019	Thomas Noël
0001-api-add-a-hashed_password-attribute-for-user-api-354.patch	5,9 ko	04 septembre 2019	Nicolas Roche
0001-api-add-a-hashed_password-attribute-for-user-api-354.patch	6,89 ko	05 septembre 2019	Nicolas Roche