# Calebasse - Bug #4808

## Permettre la modification en masse des mots de passe ou mettre en oeuvre une stratégie

13 mai 2014 12:11 - Mikaël Ates (de retour le 29 avril)

| | | | |
|---|---|---|---|
| **Statut:** | Rejeté | **Début:** | 13 mai 2014 |
| **Priorité:** | Normal | **Echéance:** | |
| **Assigné à:** | | **% réalisé:** | 0% |
| **Catégorie:** | | **Temps estimé:** | 0:00 heure |
| **Version cible:** | 2.0 | | |
| **Patch proposed:** | | **Planning:** | |

**Description**

Dans un premier temps faire un script pour créer des nouveaux mots de passe pour tous les comptes actifs et créer un fichier descriptif. Puis permettre de modifier les mots de passe des comptes avec ce fichier. CEla dans le but de pouvoir informer les utilisateurs que leur mot de passe va être modifié à une date ultérieur.

Dans un second temps il faudra permettre de compléter cela avec une stratégie de mot de passe imposant le renouvellement périodique ou a une date donnée, avec une complexité de mot de passe.

### Historique

#### #1 - 13 mai 2014 12:22 - Frédéric Péters

C'est sans doute déjà ça qui est prévu mais je pose quand même la question, l'idée c'est d'avoir un module indépendant, qui puisse être également utilisé dans authentic ?

De manière générale, je suis récemment tombé sur la documentation de FreeIPA, la partie sur les mots de passe peut nous donner une idée de ce qui peut être implémenté : http://docs.fedoraproject.org/en-US/Fedora/15/html/FreeIPA_Guide/user-pwdpolicy.html

#### #2 - 17 juin 2014 10:16 - Benjamin Dauvergne

Le premier paragraphe n'est pas bien clair pour moi, le but serait de générer un fichier de mot de passe avant leur modification effective pour pouvoir le transmettre aux utilisateurs auparavant pour qu'ils ne soient pas surpris du changement ?

Ça a priori c'est facilement faisable avec un truc comme ça:

```
from django.contrib.auth.hashers import make_password
from django.contrib.auth.models import User

import json
import random
import csv

data = []
csv = [('username', 'password')]

def generate_password():
    pass # génère un mot de passe avec les règles qu'il faut

for user in User.objects.all():
   pwd = generate_password()
   csv_data.append((user.username.encode('utf-8'), pwd))
   data.append({
      'pk': [user.username],
      'fields': {
          'password': make_password(pwd),
       },
   })

with file('new_passwords.json', 'w') as f:
    json.dump(data, f)

with file('new_passwords.csv', 'w') as f:
    writer = csv.writer(f)
    writer.writerows(csv_data)
```

On envoie le CSV aux personnes au contact des utilisateurs et le jour J on charge le fichier json avec manage.py loaddata new_passwords.json.

En matière de politique de mot de passe on a déjà l'embarras du choix entre les RFCs LDAP, la documentation d'Active Directory ou celle de Freelpa qu'a trouvé Fred, on trouvera toujours un truc à implémenter. Ce qu'il faut au niveau de Django:

- stocker/récupérer les informations supplémentaires liés à la politique du mot de passe, comme la date de dernière modification ou la listes des n derniers mots de passe pour empêcher la réutilisation,
- forcer des actions sur l'utilisateur comme changer son mot de passe, mais il y a d'autre actions intéressante comme forcer la validation d'un message de service/acceptation d'un nouveau règlement (vu sur le LemonLDAP de la Gendarmerie). On peut soit utiliser un middleware (ça bouffe un poil en perf) soit fournir une page de login qui gère ça (c'est très intrusif, pour authentic par exemple qui modifie déjà la page de login pour d'autres raisons).

Donc pour moi il faut déjà éclater ce ticket en deux.

#### #3 - 20 juin 2014 16:13 - Mikaël Ates (de retour le 29 avril)

*- Version cible mis à 2.0*