

Password Cracker (Casseur de mots de passe)

Description

Programme qui permet de cracker (casser) des mots de passes encryptés selon une fonction de hachage. L'utilisateur doit fournir un fichier texte contenant la liste des utilisateurs ainsi que leur mot de passe haché, formaté comme suit :

```
bob:8b433670258f79578f9a4e5ea388b007
jean:08da50bd109c7fb1bec49d15ae86e55f
claude:a8f6830bce790a8a67fc2e84e12093ba
superuser007:a1234b3161b4fbfdfb96dd576b65bbea
awesomeMan:6d4db5ff0c117864a02827bad3c361b9
superCoolMan:9460370bb0ca1c98a779b1bcc6861c2c
```

Format : utilisateur:motDePasseHaché

On peut aussi spécifier le caractère séparateur manuellement (voir [Maquette #1](#)).

L'utilisateur a ensuite 3 choix :

- Fournir un dictionnaire de mots
 - Un fichier texte avec un mot par ligne. Il y a de nombreux dictionnaires de disponibles en ligne.
- Fournir une Rainbow Table (table arc-en-ciel) *[J'ai des doutes qu'on puisse être capable de faire ça et de rendre ça optimisé]*
 - C'est une liste de toutes les combinaisons possibles d'un certain nombre de caractères, avec leur hash.
 - Par exemple, pour un mot de passe en 1 et 10 caractères utilisant des lettres minuscules (a-z) ou un chiffre (0-9) (ou les deux) il y a **3,760,620,109,779,060** combinaisons possibles. La table arc-en-ciel pèserait environs 316 Go! *[Comment lire ça de manière optimisée?]*.
- Brute Force (essayer toutes les combinaisons possibles)
 - On peut spécifier les caractères qu'on veut tester (Ex. : toutes les lettres majuscules, toutes les lettres minuscules, longueur max, liste de caractères spéciaux, etc.)
 - Le tout serait présenté en checkbox (on coche les options qu'on veut)
 - L'utilisateur pourra aussi fournir la liste des caractères qu'il voudrait inclure dans le brute force
 - C'est moins optimal qu'une table arc-en-ciel, puisqu'on doit générer nous-même les différentes combinaisons, mais au moins l'utilisateur n'a pas besoin de fournir un fichier de 316 Go! Ici, la vitesse dépend beaucoup du processeur de l'utilisateur.

L'utilisateur clique ensuite sur un bouton « Crack All the Passwords! » et attends...

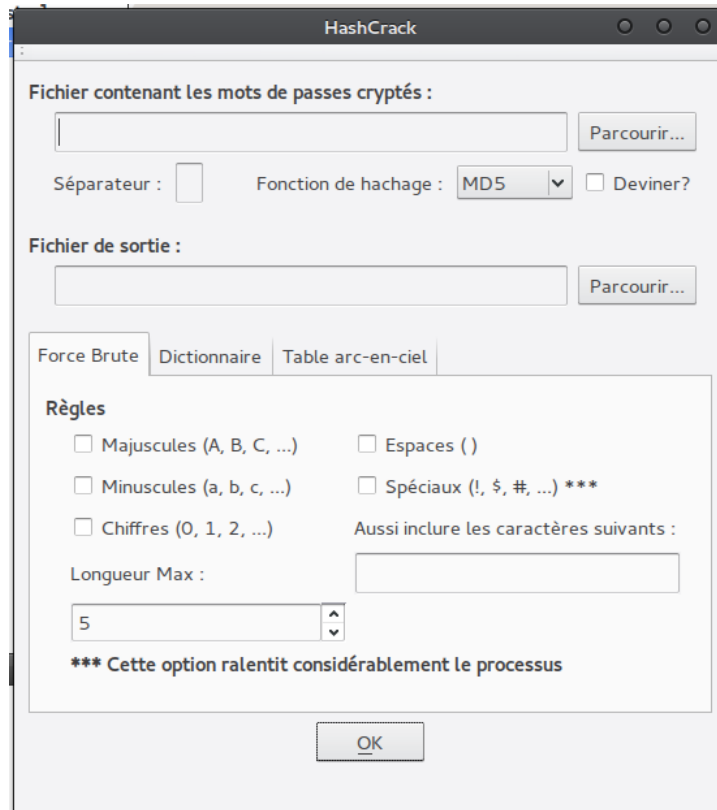
Il serait intéressant d'avoir une fonctionnalité qui permet de générer une table de correspondance (Lookup Table) entre les mots d'un dictionnaire et son hash. Essentiellement, l'utilisateur fournit une liste de mots, et le programme s'occupe de calculer le hash de chaque mot, et l'enregistre en mémoire. Ensuite on les trie, et finalement on exporte le tout dans un fichier texte. L'utilisateur peut ensuite utiliser ce fichier texte pour faire une attaque par dictionnaire.

Objectifs

- MVC
 - L'interface
 - Les fonctions de hachages et autres algorithmes
 - Le projet « Casseur de mot de passes » (Password Cracking).
 - Les tests
- Multithreading
- Implémentation d'au moins 2 fonctions de hachage nous-mêmes (MD5 et SHA-1 sont relativement simples. Il serait intéressant d'avoir aussi le SHA-256).
- Architecture modularisée. C'est-à-dire que le programme est facilement extensible. Par exemple, ajouter une fonction de hachage ne devrait pas changer toutes les classes du programme. Ça veut donc dire : utilisation d'interfaces abstraites.
- Il y a beaucoup de potentiel au niveau des tests. Il y a plusieurs choses à tester.
- C'est un bon travail d'équipe. MVC simplifie la tâche un peu.

Maquette #1

Maquette fait avec QT Designer



The image shows a Qt Designer window titled "HashCrack". The window contains a form for password cracking. It has a title bar with standard window controls. The main area is divided into several sections. At the top, there's a label "Fichier contenant les mots de passes cryptés :" followed by a text input field and a "Parcourir..." button. Below this is a "Séparateur :" label with a small square input field, a "Fonction de hachage :" label with a dropdown menu set to "MD5", and a "Deviner?" checkbox. The next section is "Fichier de sortie :" with another text input field and a "Parcourir..." button. Below this is a tabbed interface with three tabs: "Force Brute", "Dictionnaire", and "Table arc-en-ciel". The "Force Brute" tab is selected. Inside this tab, there's a "Règles" section with four checkboxes: "Majuscules (A, B, C, ...)", "Minuscules (a, b, c, ...)", "Chiffres (0, 1, 2, ...)", and "Espaces ()". There's also a checkbox for "Spéciaux (!, \$, #, ...) ***". To the right of these is a label "Aussi inclure les caractères suivants :" followed by a text input field. Below the checkboxes is a "Longueur Max :" label with a spin box set to "5". At the bottom of the "Règles" section is a warning message: "*** Cette option ralentit considérablement le processus". At the very bottom of the window is an "OK" button.

HashCrack

Fichier contenant les mots de passes cryptés :

Parcourir...

Séparateur : ☐ Fonction de hachage : MD5 ☐ Deviner?

Fichier de sortie :

Parcourir...

Force Brute Dictionnaire Table arc-en-ciel

Règles

☐ Majuscules (A, B, C, ...) ☐ Espaces ()

☐ Minuscules (a, b, c, ...) ☐ Spéciaux (!, \$, #, ...) ***

☐ Chiffres (0, 1, 2, ...) Aussi inclure les caractères suivants :

Longueur Max :

*** Cette option ralentit considérablement le processus

OK

Notes

Attaque par dictionnaire:

Lorsque l'utilisateur fournit son dictionnaire de mots, on va lire le fichier en mémoire, haché tous les mots avec la fonction de hachage voulue (et mettre ça dans une map, ou dans le même fichier dictionnaire).

Ex : Disons que mon dictionnaire (mots.txt) contient les mots

sausage

blubber

pencil

Ce qu'on ferait, c'est qu'on ouvrirait mots.txt, et on calculerait le mot haché à côté du mot, donc ça donnerait :

sausage 8b433670258f79578f9a4e5ea388b007

blubber 08da50bd109c7fb1bec49d15ae86e55f

pencil a8f6830bce790a8a67fc2e84e12093ba