LSINF1252

Password Cracker: Architecture

Groupe 8

Marques Mathias (13181700) - Grognard Simon (37811700)

1 Avril 2019

1 Introduction

Dans le cadre du cours de système informatique I (LSINF1252) il nous a été demandé d'implémenter un programme permettant de cracker des mots de passe qui ont été hashé au préalable. Nous devons renvoyé les mots de passe contenant le plus de voyelle ou de consonne. Avant de commencer la programmation nous avons choisi une certaine architecture pour notre code. qui se trouve dans ce document.

2 Structures de données

Notre structure de données sera composée des mots de passes hashés en binaire de 32 bytes de long. Nous allons les regrouper par byte (=groupement de 8 bits) et ainsi les mettre sous formes hexadécimale. Exemple de conversion d'un byte en hexadécimale : si nous prenons le byte '01111011' cela nous donne en hexadécimale '7B' en mettant tout ces nombres hexadécimaux ensemble nous obtenons la forme textuelle du hash que nous stockerons sous la forme d'un tableau de caractère.
Ensuite, grâce à ces hash textuels, nous pourrons les convertir en un autre tableau de caractère qui forme le mot de passe décrypter, à l'aide de la fonction 'reversehash' donnée dans le cadre du projet Password Cracker. Pour avoir une certaine facilitée par la suite, nous stockerons ces mots de passes dans un tableau à une entrée ainsi qu'un second tableau contenant les longueurs des mots associés. Ainsi nous pourrons calculer le nombre d'occurrences des voyelles et des consonnes que nous disposerons dans deux tableaux distincts pour extraire la valeur maximum de ces deux tableaux et enfin obtenir les mots de passes attendus par l'état que nous renverrons sur la sortie standard (stdout).

3 Type de threads

Le type de thread que nous utiliseront est le thread POSIX. Dans un premier temps nous décomposerons le fichier d'input en groupe de 32 bytes. Ensuite les threads entre en jeu, chacun d'entre eux prendra en argument un des groupes de 32 bytes. Et convertira

l'écriture binaire en hash textuel en suivant l'exemple donné dans la section *structure de données*. Ce hash textuel sera convertit en un tableau de caractère à l'aide de la fonction "reversehash" donnée, qui par la même occasion nous renverra la longueur du mot de passe. Après cela nous analyserons le nombre de voyelle et de consonne dans les mots afin de trouver ceux avec la plus grande occurrence de voyelle ou de consonne comme expliqué ci-dessus (*structure de donnée*). Ce nombres de voyelle et consonne sera retourné dans la méthode main à l'aide de pointeur créé au préalable. Ce qui est l'unique utilisation des threads dans notre programme. Pour finir, ceux-ci seront détruits.

4 Méthode de communication entre les threads

Afin que nos divers threads puissent s'exécuter sans le problème d'exclusion mutuelle, nous utiliserons les **mutex**. Ces derniers seront utilisé dans la seul section critique de notre code. C'est à dire le moment où l'on va stocker les divers résultats obtenus (voir les sections précédente) dans leur tableaux respectif qui seront pointé par un pointeur mis en argument dans les divers threads qui auront été créé.

5 Information communiquées entre les threads

Dans notre programme, la seule variable qui sera communiquée entre les threads est une variable globale qui permettra de placer les résultats obtenus (voir les sections précédente) dans les différents tableaux à leur bon emplacement. Pour ce faire après avoir mis les résultats dans les tableaux cette variable sera incrémentée afin de pouvoir placé l'élément suivant au bon endroit dans les tableaux de donnée.