TP 1 - Sécurité informatique

Arthur Coruble, Lucien Le Roux

Numéro de sujet: 2

Le but de ce TP est de casser une liste de hashes MD5 pour retrouver les mots de passe originaux. Pour ce faire, nous avons écrit un programme C++ qui gère à la fois les attaques par dictionnaire et par énumération.

1. Cassage de mots de passe

1. Attaque par dictionnaire

Le principe est simple: on lit depuis un fichier une liste de mots de passe potentiels. Pour chaque mot de passe, on calcule le hash MD5 correspondant et on compare celui fourni en entrée. Si les deux correspondent, alors nous avons trouvé le mot de passe. Cette attaque est assez rapide, et le temps d'exécution est proportionnel à la longueur du dictionnaire. Nous avons cassé les hashes suivants:

- bd787548b57d899d882ae21ca8e75263 -> acerera
- 1ed001839e6709c59e1ca1357952d8f5 -> barrons
- 27eacbc30f7fde770cc274bdb20bc186 -> beants
- e0be4fd60faa2f3d904dc237eb2647a9 -> achevas
- 3961a903097f9cb79f07dae17bac2f11 -> trainee
- 4130a0993d4c2e9486768e887104b980 -> progres
- 6268dda92c28ed9a39024fc6cfd360e9 -> cedule
- 38ffed95337f0cd8a1870ac111533671 -> repoudra
- 7994f0b70ac711650b7c6a6e2ab47c65 -> ecobues
- beffd790d7dedae2a68d507932b2732c -> deboursa

2. Attaque par énumération

L'attaque par énumération consiste à générer toutes les chaînes de caractères possibles étant donné un jeu de caractères. Ainsi, pour l'alphabet "abcd", l'algorithme devra générer a, b, c, d, aa, ab, ac, ad, ba, bb, et ainsi de suite. La génération de nouvelles chaînes continue jusqu'à ce que l'on en trouve une dont le hash corresponde à celui donné en entrée. En l'occurrence, les caractères possibles sont:

`"abcdefghijklmnopqrstuvwxyz0123456789!@#$%&*"` (43 caractères)

Ainsi, pour une chaîne de 8 caractères, il existe 43^8 possibilités, ce qui est extrêmement long et coûteux à calculer. Nous avons donc tiré profit du multithreading, mais les performances restent liées aux processeurs de la machine. Nous avons donc pu trouver les hashes suivants:

- 7635a53a1e8308e8452a33cc1595316a -> $3j@g3$
- c69659150abcc53287dd263361b86077 -> 1odlons
- b9a6425b8b5b41a8a0d56d1e6fbf4b95 -> p4$3nt4l
- 72aa6cfbe3f8a385e8391303eac0f34f -> c4q@3tt3
- 2716c1e5bee3a6d31e720aab9d769a8d -> s@c$3$13

2. Sur la complexité de casser des mots de passe

Les résultats précédents nous permettent de conclure que le choix d'un mot de passe est crucial et que ce dernier doit être robuste afin d'éviter de se le faire voler. Afin de se protéger des attaques par dictionnaire et bruteforce basiques, il ne faut surtout pas choisir un mot de passe qui correspond à un mot commun. Il faudrait utiliser la substitution de caractères (bien que la plupart des casseurs de mots de passe prennent en compte ces substitutions, telles que `a = 4` ou `e = 3`, elles sont autant de combinaisons à essayer et donc le cassage prendra plus de temps). Il faut également altérer le mot de base, en ajoutant ou modifiant certaines lettres par exemple. Par ailleurs, il faut éviter de choisir un mot de passe se rapportant à une information personnelle telle que la date de naissance, le nom de son chien, … car le social engineering pourrait permettre de le trouver aisément. De plus pour ce qui est de l'attaque par énumération, nous ne pouvons que recommander l'utilisation d'un mot de passe assez long (8 caractères et plus) et qui dispose d'un jeu de caractères étendu (majuscules, caractères spéciaux, chiffres) car cela augmentera énormément le nombre de combinaisons possibles et donc réduit les chances de casser le mot de passe en un temps raisonnable. Une stratégie simple pour choisir un mot de passe serait donc d'utiliser des caractères variés, et qu'il soit assez long. Afin de le retenir plus facilement, on peut essayer de calquer le mot de passe sur un mot de base (peu commun, n'apparaissant pas dans un dictionnaire) ou sur un autre moyen mnémotechnique (par exemple, en suivant une disposition particulière de touches du clavier, bien qu'encore une fois, certains casseurs prennent en compte cette possibilité).

3. Le programme

Notre programme est écrit en C++ et gère les attaques par dictionnaire et par énumération. Nous utilisons les librairies suivantes:

- OpenSSL qui fournit une fonction MD5 pouvant hasher une chaîne de caractères
- OpenMP qui nous permet de paralléliser notre attaque par énumération afin de l'accélérer.

Ces dernières doivent donc être présentes sur la machine compilant le code. Le build se fait sous Linux via un Makefile.

Le programme s'utilise comme suit:

```
./crack [-d dictionary] [digest…]
```

où `digest` est une suite de hashes MD5 valides. Les arguments sont parsés par l'utilitaire getopt. Selon les flags, le programme suivra différentes stratégies:

- Dictionnaire: L'attaque par dictionnaire est utilisée, le chemin vers le dictionnaire est passé en paramètre

Pour accéder à la liste des options, utilisez le flag `-h`. Par défaut, l'attaque par énumération est toujours activée. Si l'on ajoute l'attaque par dictionnaire, celle-ci sera utilisée avant celle par énumération car elle est plus rapide.

Le code a été conçu de manière à pouvoir étendre les possibilités dans le futur: ajout de nouvelles attaques, possibilité de casser des hashes autres que MD5, … afin de devenir un outil de cassage de mot de passe plus utile.