> diminuer, il est sage de la prendre en compte pour des mots de passe qu'on veut sécuriser durablement.

Pour un mot de passe vraiment robuste au sens de l'Anssi, il faudrait par exemple demander une suite de 16 caractères pris chacun dans un ensemble de 200 caractères. Cela ferait un espace de 123 bits.

Ce n'est guère envisageable, aussi les concepteurs de systèmes sont-ils moins exigeants et se contentent-ils d'espaces de mots de passe faibles ou moyens; ils n'appliquent l'exigence de clés longues que lorsque l'utilisateur n'a pas à la mémoriser lui-même, car elle est gérée automatiquement par le système.

D'autres procédés sont mis en œuvre pour empêcher l'exploration de l'espace des possibilités et casser les mots de passe. Le plus simple est bien connu et utilisé par les cartes bancaires: au bout de trois essais infructueux, la carte est bloquée. D'autres idées ont été proposées comme doubler les temps d'attente entre deux essais à chaque nouvelle erreur, sauf s'il s'est passé un long moment, 24 heures par exemple, auquel cas le système se remet à zéro.

Ces méthodes sont cependant inopérantes quand l'attaquant du système peut travailler hors ligne et mener des essais secrètement de manière isolée, ou que le système ne peut pas être configuré pour arrêter et freiner les essais infructueux.

Assez souvent, un attaquant réussit à se procurer le mot de passe chiffré ou l'empreinte du mot de passe (nous allons voir ce qu'est une empreinte) en accédant à des données du système qu'il attaque. Il peut alors, en toute tranquillité, pendant des jours entiers ou même des semaines, mener toutes les tentatives qu'il souhaite pour casser les mots de passe dont il s'est emparé sous forme chiffrée.

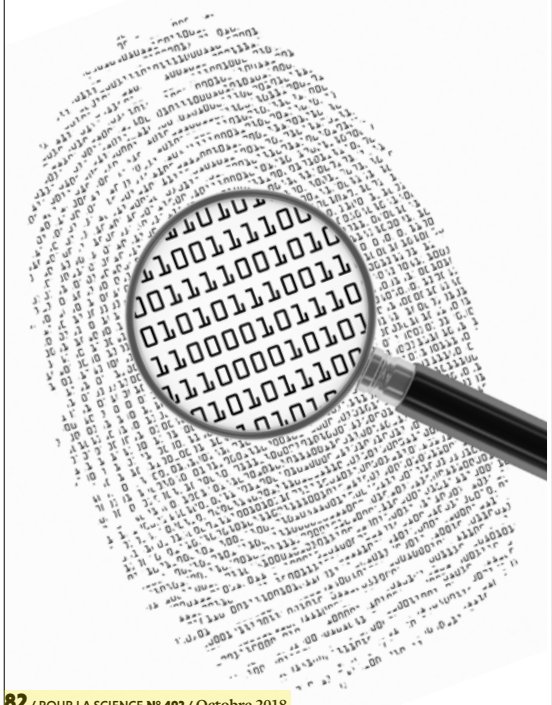
Avant d'expliquer les subtils procédés utilisés par de tels attaquants, revenons sur la taille de l'espace des possibilités.

Quand nous avons évalué la taille en bits d'un espace de clés ou mots de passe, dénommée entropie, nous avons implicitement supposé que l'utilisateur choisit au hasard uniformément son mot de passe dans cet espace. Le plus souvent, ce n'est pas le cas, car pour mémoriser un mot de passe on préférera un mot courant, par exemple «locomotive», à un mot vraiment quelconque, par exemple «xdichqewax».

DES DICTIONNAIRES POUR ATTAQUER

C'est là un grave problème qui donne lieu aux attaques par dictionnaires. Des listes de mots de passe couramment utilisés ont été collectées et classées en fonction de la fréquence avec laquelle ces mots de passe sont utilisés. Un attaquant tentera des essais en prenant l'une de ces listes et en l'utilisant dans l'ordre.

2 LES PIRATES DÉÇUS PAR LES FONCTIONS DE HACHAGE



Les fonctions de hachage cryptographique sont des procédés soigneusement mis au point qui transforment un fichier informatique F, aussi long soit-il, en une suite *h*(F) de petite taille appelée empreinte de F. Par exemple, la fonction notée SHA256 transforme la phrase « Il fait beau » en :
316FF650DC5FFC6F8B9CFF25069942F4AA5088
78930A1410D81D5F1DD8C71077
(64 caractères hexadécimaux qui sont équivalents à 256 bits).

Changer un seul caractère du fichier change totalement son empreinte. Par exemple, pour « il fait beau », la fonction SHA256 donne l'empreinte :
D66D08EC93EBA6ACCDD552E2A38EBE6474285
E0D4F2370A87E206553E29AB93C.

Vous pouvez refaire et vérifier ces calculs à l'adresse :
https://passwordsgenerator.net/sha256-hash-generator/
ou https://www.xorbin.com/tools/sha256-hash-calculator.

Les bonnes fonctions de hachage produisent des résultats analogues à ceux qui seraient obtenus s'ils étaient tirés uniformément au hasard. Surtout, pour un résultat possible arbitraire (une suite de 64 caractères hexadécimaux), il est impossible de trouver en un temps raisonnable un fichier F ayant cette empreinte.

Il existe plusieurs générations de fonctions de hachage. Les générations SHA0 et SHA1 sont considérées comme obsolètes et déconseillées. La génération SHA2, à laquelle appartient SHA256, est considérée comme sûre.

Au lieu de mémoriser les mots de passe de leur client, les serveurs internet doivent mémoriser les empreintes de ces mots de passe. En cas d'intrusion, il sera alors impossible ou très difficile au pirate informatique d'exploiter ce qu'il trouve.

Cela fonctionne remarquablement bien car, sans contraintes particulières, nous sommes tous tentés de choisir des mots simples, des noms, des prénoms, des petites phrases, et que les possibilités sont alors relativement peu nombreuses.

Cette utilisation non uniforme des possibilités équivaut à une réduction de l'espace des possibilités, ce qui diminue le nombre moyen d'essais pour casser un mot de passe.

Voici les 25 premiers éléments d'un de ces dictionnaires de mots de passe. Elle est tirée d'une base de 5 millions de mots de passe ayant fuité en 2017 et repérés par SplashData:

1. 123456; 2. Password; 3. 12345678; 4. qwerty; 5. 12345; 6. 123456789; 7. letmein; 8. 1234567; 9. football; 10. iloveyou; 11. admin; 12. welcome; 13. monkey; 14. login; 15. abc123; 16. starwars; 17. 123123; 18. dragon; 19. password; 20. master; 21. hello; 22. freedom; 23. whatever; 24. qazwsx; 25. trustno1.

Remarquez le «password» en seconde position et le sympathique «iloveyou» en position 10. Bien sûr, des telles listes changent selon le pays où elles sont collectées, les sites concernés et au cours du temps.

Pour les mots de passe de 4 chiffres (par exemple le code PIN des cartes SIM des téléphones), les résultats sont encore plus étonnants. En 2013, le site DataGenetics, s'appuyant sur une collecte de 3,4 millions de mots de passe, chacun comportant 4 chiffres, a mesuré que la suite de 4 chiffres la plus utilisée était 1234, qui représentait 11% des choix, suivie par 1111 et 0000 avec des scores de 6% et 2%.

Le mot de passe de 4 chiffres le moins utilisé était 8068. Méfiez-vous quand même, ce n'est peut-être plus vrai maintenant que ces résultats ont été publiés. Le choix 8068 n'était présent que 25 fois parmi les 3,4 millions de suites de 4 chiffres de la base, ce qui est beaucoup moins que les 340 utilisations qu'on aurait trouvées si chaque combinaison de 4 chiffres avait été utilisée avec la même fréquence.

Les 20 premières séries de 4 chiffres sont: 1234; 1111; 0000; 1212; 7777; 1004; 2000; 4444; 2222; 6969; 9999; 3333; 5555; 6666; 1122; 1313; 8888; 4321; 2001; 1010.

Même sans dictionnaire de mots de passe, l'utilisation des différences entre fréquences d'usage des lettres (ou des doubles lettres) dans une langue permet d'organiser efficacement une attaque. Certaines méthodes d'attaque prennent aussi en compte que, pour faciliter la mémorisation, on choisit des mots ayant une certaine structure comme A1=B2=C3, AwX2AwX2, O00.Ili. (que j'ai longtemps utilisé), ou alors obtenus en combinant plusieurs mots simples comme password123 ou johnABC0000. En exploitant de telles régularités, on détermine un ordre d'attaque des mots de passe qui accélère la recherche. La conclusion est simple: dans un espace donné, il faut vraiment choisir au hasard son mot de passe. Certains logiciels le font pour vous et vous proposent donc un mot de passe au hasard. Il faut cependant être conscient qu'un logiciel qui engendre un mot de passe au hasard risque d'utiliser, délibérément ou non, un mauvais générateur pseudoaléatoire, auquel cas ce qu'il propose sera imparfait.

Informés de tout cela, les sites internet convenablement conçus analysent les mots de passe proposés au moment de leur création et refusent ceux qu'il serait trop facile de retrouver. C'est agaçant, mais c'est pour votre bien!

TESTER LES MOTS DE PASSE...

Un outil permet de vérifier si l'un de vos mots de passe n'a pas déjà été piraté. Il utilise une base de plus de 500 millions de mots de passe obtenue en regroupant des listes dévoilées à la suite d'attaques diverses:

https://haveibeenpwned.com/Passwords

J'ai essayé e=mc2e=mc2 que j'aimais bien et pensais sûr, et j'ai obtenu la réponse déconcertante: «This password has been seen 110 times before.» Quelques autres essais montrent qu'il est difficile de proposer un mot de passe facile à mémoriser et que la base ne connaît pas. Par exemple, aaaaaa est apparu 353071 fois; a1b2c3d4, 105134 fois; abcdcba, 353 fois; abczyx: 158 fois; acegi, 111 fois; chirac, 600 fois; sarkozy, 680 fois; hollande, 832 fois; macron, 279 fois.

Notons qu'être original reste possible. Voici cinq exemples de mots de passe que le site n'a pas dans sa liste: eyahaled (mon nom à l'envers); bizzzzard; meaudepace; modeuxpass; abcdef2018. Maintenant que je les ai essayés, je me demande si la base ne pourrait pas les ajouter lors de sa prochaine mise à jour et je ne les utiliserai donc pas!

Le NIST (National Institute of Standards and Technology), aux États-Unis, a récemment publié un avis où il recommande la méthode des dictionnaires pour filtrer le mot de passe choisi par un utilisateur qui est en train d'en créer.

... ET SAVOIR LES CONSERVER

Parmi les règles qu'un bon concepteur de serveur internet doit absolument appliquer, il y a celle-ci: ne pas garder dans les mémoires de l'ordinateur qui fait fonctionner le site internet la liste en clair des couples [identifiant d'utilisateur, mot de passe].

La raison est évidente: un pirate pourrait accéder à l'ordinateur contenant cette liste, soit parce que le site est mal protégé, soit parce qu'une faille profonde dans le système, ou dans la puce au cœur de la machine, aura été exploitée, alors qu'elle est inconnue de tous (faille Zero-day)... sauf de l'attaquant. >

BIBLIOGRAPHIE

Wikipedia, **Mot de passe** et **Rainbow table**, entrées consultées en juin 2018.

G. Avoine *et al.*, **How to handle rainbow tables with external memory**, dans J. Pieprzyk et S. Suriadi (éd.), *Information Security and Privacy - ACISP 2017*, pp. 306-323, Springer, Cham, 2017.

P. A. Grassi *et al.*, **Digital Identity Guidelines**, NIST Special Publication 800-63-3, 2017 (https://doi.org/10.6028/NIST.SP.800-63-3).

G. Avoine *et al.*, **Characterization and improvement of time-memory trade-off based on perfect tables**, *ACM Transactions on Information and System Security*, vol. 11(4), article 17, 2008.

P. Oechslin, **Making a faster cryptanalytic time-memory trade-off**, dans D. Boneh (éd.), *Advances in Cryptology - CRYPTO 2003*, pp. 617-630, Springer, 2003.

M. E. Hellman, **A cryptanalytic time-memory trade-off**, *IEEE Transactions on Information Theory*, vol. IT-26, n° 4, pp. 401-406, 1980.