Mots de passe

Table des matières

1 Mots de passe

1.1 Mots de passe

Un mot de passe est un secret, généralement choisi par l'utilisateur, permettant d'authentifier un utilisateur.

1.2 Durcissement des mots de passe

Le mot de passe pouvant être l'unique barrière entre un attaquant et un système, il est important que le mot de passe soit suffisamment costaud. Comme sur de nombreux systèmes c'est l'utilisateur qui choisi un mot de passe, il est nécessaire que le système s'assure que le mot de passe choisi ne soit pas trop simple.
Pour cela à chaque fois qu'un utilisateur choisi un nouveau mot de passe, le système vérifie que le mot de passe suit la politique des mots de passe décidée dans l'entreprise. Généralement un mot de passe doit avoir des propriétés adéquates en fonction : de la longueur minimale, par exemple ou moins 8 caractères de la diversité des caractères le composant, par exemple au moins une majuscule, une minuscule, un chiffre et un caractère autre et optionnellement de sa non dérivabilité depuis un mot d'un dictionnaire.

1.3 Mots de passe à usage unique

Il est possible de mettre en place un système de mots de passe à usage unique. Ceci empêche un voleur, qui aurait capté votre mot de passe sur un réseau ou qui vous aurait vu le taper, de le réutiliser.
Le système génère une liste de mots de passe qui est fournie à l'utilisateur. L'utilisateur utilise le premier mot de passe non utilisé de la liste. Quand la liste a été entièrement utilisée, le système en génère une autre.
Une façon de générer cette liste est de générer des mots de passe aléatoires n'ayant aucun lien de l'un à l'autre.
Une façon moderne de générer cette liste est d'utiliser un secret principal et d'en dériver une liste de mots de passe en passant via une fonction de hachage le secret principal, puis tous les hachés obtenus les uns après les autres. La liste fournie à l'utilisateur est la liste de ces résultats mais à l'envers, chaque mot de passe dépendant du mot de passe suivant. Ceci permet qu'un mot de passe ne dépende pas d'un mot de passe précédent déjà utilisé. Un attaquant ayant capturé plusieurs mots de passe n'est pas capable de facilement en déduire le mot de passe suivant.

1.4 Passphrase

Une passphrase, ou phrase de passe, est un mot de passe étendu. Il est caractérisé par l'utilisation de plusieurs mots séparés par des caractères non alphabétiques. Si les mots sont suffisamment complexes et indépendants les uns des autres, et que les caractères non alphabétiques sont suffisamment nombreux et diversifiés alors une passphrase est beaucoup plus complexe à casser tout en pouvant être plus facile à retenir.

2 Stockage

2.1 Chiffrement du mot de passe

Pour stocker de façon sécurisée un mot de passe la méthode naïve est d'utiliser une clé secrète pour chiffrer le mot de passe. Lorsque l'utilisateur souhaite s'authentifier il faut alors déchiffrer le mot de passe et le comparer à celui proposé par l'utilisateur.
Le problème est alors de stocker la clé secrète de façon sécurisée car si quelqu'un l'obtient il est alors capable de déchiffrer tous les mots de passe. Cette clé secrète ne peut pas être stockée en clair car elle sera trouvée. Si elle est stockée de façon chiffrée avec une clé secrète secondaire alors elle aura le même problème de stockage de façon sécurisée que la clé secrète principale. Même en imaginant cela possible, lors de l'utilisation d'une clé secrète celle-ci se trouve en mémoire et peut donc être récupérée.

2.1.1 Hachage du mot de passe

Une façon de stocker un mot de passe est d'utiliser une fonction à sens unique, ie une fonction qui ne possède pas de fonction inverse qui à partir d'une image de mot de passe permet d'obtenir rapidement le mot de passe en clair. Il existe de telles fonctions à sens unique nommées fonctions de hachage telles que MD5, SHA1, SHA2… Le mot de passe d'un utilisateur est alors « haché » et stocké dans une « base » à l'abri des utilisateurs.
Note : même si ce n'est pas exact, l'expression utilisée pour désignée l'image du mot de passe enregistrée dans la base est « mot de passe chiffré ». Ceci est un abus de langage puisqu'en aucun cas il est chiffré. S'il était chiffré alors il existerait une fonction pour récupérer le mot de passe en clair à partir du mot de passe « chiffré », ce qui n'est pas le cas par définition.
Pour vérifier le mot de passe d'un utilisateur, le mot de passe proposé par l'utilisateur est haché et comparé à la version « chiffrée » enregistrée dans la base. Si l'image ainsi obtenue est égale à celle enregistrée dans la base alors le système considère que le mot de passe proposé est celui de l'utilisateur qui est alors authentifié.
Avec cette méthode il existe une possibilité d'existence de mots de passe qui ont la même image que le mot de passe enregistré par l'utilisateur initialement. Ceci est nommé une collision et les fonctions de hachage sus-cité ont la propriété de rendre extrêmement difficile de trouver des collisions, notamment quand l'image est donnée d'avance.
Aujourd'hui la meilleure attaque sur les fonctions de hachage classiques (ici limitées à MD5 et SHA1) est d'arriver à trouver deux textes en clair qui possèdent la même image, mais ceci nécessite de partir d'un texte en clair et d'en modifier une partie de deux façons différentes jusqu'à trouver deux textes différents en parties mais avec une même image. A chaque étape l'image des deux textes diffèrent des images de l'étape précédente et il n'est pas possible même avec cette méthode de trouver un texte en clair avec une image donnée d'avance.

2.2 Cassage de l'empreinte d'un mot de passe

2.2.1 Cassage par force brute

Si un attaquant a accès aux mots de passe « chiffrés », par exemple après avoir piraté un système, il peut alors tenter de calculer les images de mots de passe possible jusqu'à trouver un mot de passe donnant la même image que celle enregistrée dans le système.
Il est ici nécessaire de « chiffrer » chaque mot de passe testé avant de le comparer au mot de passe « chiffré » à casser.
L'attaquant peut tester toutes les combinaisons possibles de chiffres, de lettres et de caractères autres pour des longueurs réduites. Cette méthode est nommée « recherche par brute force hors ligne », car il peut se déconnecter du système piraté une fois les mots de passe « chiffrés » volés, ou simplement brute force.
Une méthode améliorée est surnommée brute force intelligente, où la génération des mots de passe possibles est générée en suivant des statistiques générées à partir d'un dictionnaire (français, informatique…) ou de mots de passe déjà cassés. L'algorithme de génération génère alors des mots de passe possibles qui ressemblent aux mot contenus dans ces dictionnaires.

2.2.2 Cassage par dictionnaire

Avant le cassage en force brute, l'attaquant va généralement commencer par prendre les mots d'un dictionnaire, général (français, anglais…) ou spécifique (informatique, biologie, science fiction…) puis leur appliquer éventuellement des règles simples de modifications (post/pré-fixer d'un chiffre, mettre une majuscule, écrire en « 31337 »…).
Si un attaquant exécute cela sur plusieurs systèmes successivement il va alors « chiffrer » plusieurs fois les mêmes mots de passe potentiels. Il est alors possible de ne chiffrer qu'une fois ses mots de passe potentiels et de les enregistrer dans une base nommé « dictionnaire pré-calculé ». Lors d'une prochaine attaque, il n'a alors qu'à effectuer une recherche dans cette base pour savoir instantanément le mot de passe en clair si l'image s'y trouve.

2.3 Durcissement des empreintes

2.3.1 La graine

Pour éviter les dictionnaires pré-calculés, le chiffrement peut intégrer une graine (salt) afin qu'un même mot de passe puisse être « chiffré » de différentes façons par le système. Si le nombre de graines différentes possibles est faible alors il sera toujours possible de calculer tous les dictionnaires pré-calculé pour un dictionnaire donné et de tous les stocker. Si le nombre de graines est important il n'est alors plus possible de chiffrer toutes les possibilités d'un dictionnaire important et de stocker tous les résultats.
L'algorithme de « chiffrement » avec une graine est de concaténer la graine avec le mot de passe avant d'en passer le résultat dans la fonction de « hachage ».
Afin que cette méthode soit efficace, il est nécessaire que la graine soit différente d'un mot de passe « chiffré » à l'autre. Afin de permettre la vérification d'un mot de passe il faut alors stocker en clair la graine avec le mot de passe « chiffré ». Ceci n'est pas un problème puisque connaître la graine ne permet pas d'en déduire immédiatement le mot de passe en clair.

2.3.2 Le nombre de tours de boucle

Casser un mot de passe chiffré requière d'autant plus de temps que le mot de passe à casser est complexe. Malheureusement avec la loi de Moore, la vitesse des processeurs double tous les 18 mois, sans parler des possibilités de « clustering » d'aujourd'hui, un attaquant pouvant même utiliser d'autres machines qu'il a piraté comme nœuds de son cluster.
Afin de ralentir les pirates il est alors nécessaire d'introduire une boucle dans la méthode de « chiffrement ». Ici au lieu de n'appeler la fonction de hachage qu'une seule fois, celle-ci est appelée N fois. Ce nombre N est doublé tous les 18 mois afin que l'attaque soit toujours aussi lente pour l'attaquant. Il est alors nécessaire de stocker ce nombre de boucles dans la configuration du système.

2.3.3 Format général d'une empreinte

Le format du mot de passe chiffré tel que stocké est alors basé sur le suivant :

```
NsGsC
```

où **N** est ne nombre de boucles utilisées, **G** la graine utilisée, **C** le résultat du « hachage » du mot de passe en utilisant **N** tours de boucle et la graine **G**, et **s** un séparateur. Bien sûr **N** et **G** sont en clair puisqu'il sont nécessaires au calcul et leur connaissance ne permettent pas d'obtenir rapidement le mot de passe en clair.
Sous Unix le format est celui-ci :

```
$i$N$G$C
```

où **$** est le séparateur et **i** est le numéro de l'algorithme de « hachage » utilisé.

2.4 Sous Unix

2.4.1 Historique

Historiquement les premiers mots de passe utilisés pour protéger les comptes utilisateurs étaient sous **Unix**. Les mots de passe ont été stockés en clair, mais rapidement ils ont été chiffrés avec un algorithme à clé secrète, celle-ci étant stockée « cachée » dans le code. La clé ayant été trouvée rapidement, un algorithme spécifique a été développé pour rendre plus complexe le cassage des mots de passe. Ceci était d'autant plus nécessaire que ces mots de passe « chiffrés » étaient stockés dans un fichier accessible à tout le monde en lecture : **/etc/passwd**

2.4.2 DES

Cet algorithme basé sur un algorithme de chiffrement (d'où l'abus de langage d'aujourd'hui) a la particularité d'utiliser le mot de passe comme clé de chiffrement pour chiffrer une chaîne de caractères connue. Pour déchiffrer il faut alors connaître le mot de passe et le résultat du déchiffrement serait alors la chaîne de caractères connue.
Cette méthode était basée sur l'algorithme de chiffrement DES. Celui-ci possédant une clé de chiffrement de 56 bits, le mot de passe était de facto limité à 8 caractères et 7 bits par caractères.
Afin que deux utilisateurs n'aient pas le même mot de passe « chiffré » s'ils ont le même mot de passe en clair, une graine est utilisée. Le nombre de graines différentes n'étant que de 1024 il est apparu rapidement que des dictionnaires pré-calculés étaient possibles.
Entre temps la possibilité de stocker les empreintes dans un fichier d'accès restreint en lecture est apparue. Ce fichier est généralement **/etc/shadow** . Il n'est accessible qu'à l'utilisateur **root** et nécessite donc des programmes **SUID root** pour pouvoir lire ou modifier une empreinte. Dans certains cas un groupe est autorisé à lire ce fichier, lui permettant alors d'authentifier les utilisateurs mais pas de modifier les empreintes.

2.4.3 MD5

Un autre algorithme a été créé, à la base plus pour avoir une fonction légale sans chiffrement, car à l'époque les fonctions de chiffrement étaient interdites à l'export depuis les Etats-Unis et de très nombreux pays en limitaient l'usage (ce qui est toujours le cas de nos jours où en France la cryptographie n'est libre qu'en dessous de 128 bits).
La fonction utilisée était **MD5** et le nombre de graines possibles était de plusieurs millions. Afin que le cassage ne soit pas trop rapide, un nombre de boucles suffisant était fixé, mais celui-ci était défini en dur et non modifiable (1000). La longueur maximale du mot de passe était infinie.

2.4.4 Blowfish

Un autre algorithme a été conçu pour **OpenBSD**. Celui-ci utilise un algorithme de chiffrement (**blowfish**) et fonctionne comme le précédant en utilisant un le mot de passe comme clé de chiffrement. Il possède un nombre de graines possibles beaucoup plus important et surtout un nombre de boucles configurable.

2.4.5 SHA2

Le dernier algorithme conçu pour **Linux** est basé sur SHA2. Il possède lui aussi un nombre important de graines possibles et un nombre de boucles configurable.

2.5 Sous Windows

2.5.1 LanMan (LM)

Le premier algorithme est **LM**, conçu par **IBM** pour des systèmes limités. Le mot de passe est tronqué à 14 caractères, s'il n'est pas assez long il est complété par des caractères nuls. Ce mot de passe est découpé en deux mots de passe 7 caractères chacun. Ces deux mots de passe sont modifiés pour changer toutes les lettres minuscules par des majuscules. Chaque mot de passe est utilisé comme clé de chiffrement pour chiffrer en **DES** une chaîne de caractères connus.
Le problème de cet algorithme est que le nombre de mots de passe différents possibles est restreint. De plus le cassage est accéléré par le fait qu'il n'est nécessaire que de casser deux mots de passe de 7 caractères au lieu d'un mot de passe de 14 caractères, ie les possibilités sont proportionnelles à 2^7 (128) au lieu de 2^14 (16384). Aujourd'hui il est possible de chiffrer tous les mots de passe possibles en moins d'une semaine.

2.5.2 NTLM

Le second algorithme est **NTLM**, créé par **Microsoft**. Le mot de passe est limité à 128 caractères, passé au format unicode et injecté dans la fonction de hachage **MD4**. Si un calcul **MD4** est plus rapide qu'un calcul **DES** utilisé dans l'algorithme précédent, casser un mot de passe en **NTLM** est beaucoup plus complexe puisqu'il est nécessaire d'avoir la bonne casse (majuscules/minuscules) et que le mot de passe peut être

beaucoup plus long que 7 caractères.
Pendant longtemps, les deux images **LM** et **NTLM** étaient stockées sur les serveurs ou postes clients. Quand un attaquant obtenait la base des mots de passe, il cassait alors l'image **LM**, puis calculait toutes les possibilités de majuscules/minuscules pour trouver le mot de passe en clair correspondant à l'image **NTLM**.
Si pendant longtemps les deux empreintes étaient calculées et enregistrées, actuellement l'image **LM** n'est plus calculée lors d'un changement de mot de passe et ne peut donc plus être enregistrée. Seule l'image **NTLM** est calculée et enregistrée.

3 Utilisations réseau

3.1 Sous Unix

Sous **Unix** quand un utilisateur requière de s'authentifier, il donne son mot de passe en clair. Celui-ci est transmis via le réseau au serveur. Le serveur récupère dans la base l'empreinte enregistrée et « hache » le mot de passe proposé avec les paramètres définis dans l'empreinte. L'image calculée est comparée à l'image enregistrée et l'accès est accordé si elles sont identiques.
Pendant longtemps les mots de passe étaient envoyés en clair sur le réseau, via les protocoles **telnet**, **ftp**, **pop**, **http**… Un attaquant peut alors écouter le réseau et intercepter les trames contenant le login et le mot de passe pour les rejouer plus tard à sa guise.
Aujourd'hui les protocoles sécurisés sont privilégiés tels que **ssh**, **sftp**, **imap**, **https**… Ils utilisent des algorithmes de chiffrement sécurisés pour mettre en place des tunnels sécurisés entre le client et le serveur, les mots de passe étant envoyés « en clair » dans ce tunnel chiffré, empêchant un attaquant d'intercepter le mot de passe en clair.
En cas de piratage d'un serveur et de vol de la base des empreintes, le pirate ne peut pas utiliser tels quels les mots de passe « chiffrés » et doit donc les casser avant de pouvoir se faire passer pour un utilisateur.

3.2 Sous Windows

Quand un utilisateur se connecte à un serveur, ce dernier envoie un challenge au client. L'utilisateur fourni le mot de passe en clair au client. Le client « hache » le mot de passe proposé en **LM** et/ou en **NTLM** et exécute une fonction prenant en paramètres une empreinte et le challenge envoyé par le serveur. Le client envoie le(s) résultat(s) de ses calculs au serveur qui en fait de même de son côté. Si les résultats reçus du client et calculés par le serveur sont identiques alors l'utilisateur est authentifié avec succès.
La faiblesse de cet algorithme est que si un pirate vole la base des empreintes, il lui est possible d'utiliser directement les empreintes **LM** et **NTLM** pour s'authentifier sans avoir à les casser au préalable. Il suffit simplement de modifier le client pour sauter une étape dans le protocole d'authentification réseau afin d'utiliser directement l'empreinte volée. Les hashes Windows sont des équivalents de mots de passe en clair.