Les One Time Passwords,
Mots de passe à usage unique

Étudiant : Nicolas Favre-Félix
IFIPS Info 3

Sommaire

Définition d'un système d'authentification par OTP.....3
Historique 3
Utilisation actuelle.....3
Les fonctions de hachage.....4
Définition.....4
Exemples de résultats de l'application d'une fonction de hachage :.....4
Fonctionnement général d'un système OTP.....5
Étude des systèmes existants et des problèmes posés :.....5
Solution proposée par les OTP.....5
Initialisation d'une base de mots de passe.....5
Exemple de mots de passe générés.....6
Authentification d'un utilisateur par OTP.....6
Pourquoi ce système est-il sécurisé ?.....7
Les limitations des systèmes OTP.....7
Combinaison avec SSH.....9
Ajout d'une sécurité supplémentaire : les tokens OTP.....11
Références.....11

Définition d'un système d'authentification par OTP

Un système d'authentification par OTP crée et utilise une suite aléatoire de caractères pour générer un ensemble de mots de passe destinés à l'authentification d'un utilisateur. Les mots de passe générés dérivent les uns des autres d'une manière telle qu'il est impossible de réutiliser un mot de passe ou de l'utiliser pour prédire le suivant, mais que chaque mot de passe dérive de son successeur.
Dans ce but, on utilise un générateur de nombres pseudos aléatoires, ainsi que des fonctions de hachage dites « à sens unique ».

Historique

Les mots de passe à usage unique ont fait leur apparition en février 1994, lorsque Neil Haller présente son système S/Key au *ISOC Symposium on Network and Distributed System Security*. Il en fera la RFC (*Request For Comments*) numéro 1760 en février 1995.

L'utilisation des mots de passe à usage unique se standardise avec l'apparition en février 1998 de la RFC 2289 dont le sujet est « *A One-Time Password System* ».

Son but premier est de rendre inefficace la capture de mots de passe circulant en clair sur un réseau informatique. Cette technologie n'empêche pas la capture des informations, mais rend impossible l'utilisation future de ces données, en instaurant une politique de péremption des mots de passe, utilisable une seule fois.

De nos jours, les moyens de cryptage disponibles rendent inefficaces les écoutes de réseau. La menace s'est déportée du réseau vers le poste client, où il n'est maintenant pas rare de trouver de nombreux logiciels espions. L'utilisation des OTP est aujourd'hui pratiquée lors de l'utilisation d'un poste auquel on ne peut faire confiance (cybercafé, réseau non sécurisé).

Utilisation actuelle

L'utilisation des technologies de One-Time Password est aujourd'hui assez peu courante. Le manque d'information des entreprises en est la principale raison, bien que des sociétés importantes dans le domaine de la sécurité informatique (Verisign, Microsoft) l'aient intégré à leurs produits à plusieurs reprises.

Sa facilité d'intégration, ainsi que la preuve qui a été apportée à l'inviolabilité de son principe en font un des prochains moyens de sécurisation des communications sur le poste client. Les communications sont aujourd'hui sécurisées, les technologies de cryptographie à clé publique sont de plus en plus utilisée, le point faible d'un système de sécurité se situe donc maintenant du côté du client, qui fait confiance aux technologies onéreuses mises en œuvre pour la sécurisation des liens réseaux, et néglige la sécurité de son propre poste.

Les fonctions de hachage

Définition

Une fonction de hachage, aussi appelée fonction de compression, est une fonction injective, qui transforme un ensemble de données X en un ensemble Y plus petit.
X est appelé l'ensemble des messages, Y l'ensemble des empreintes numériques. On appelle h la fonction de hachage de X vers Y. On définit x Є X, et y Є Y tel que y = h(x).

Un bon algorithme de hachage doit rendre trois problèmes difficiles à résoudre :

1. Pré image : Connaissant y, trouver x Є X tel que h(x) = y

2. Seconde pré image : Connaissant x Є X, trouver x' Є X tel que $x' \neq x$ et h(x) = h(x')

3. Collision : Trouver x et x' tels que $x \neq x'$ et h(x) = h(x')

Si aucun de ces problèmes ne peut être résolu facilement, on dit que la fonction de hachage est « à sens unique », c'est à dire qu'on considère impossible en un temps raisonnable l'inversion de cette fonction pour une empreinte y donnée.

La notion de temps est importante lors de l'utilisation de mots de passe. Le hachage d'un mot de passe et la transmission de son empreinte ne peut être possible que s'il est humainement impossible d'inverser ce hash. Les fonctions de hachage utilisées de nos jours permettent de garantir que la recherche exhaustive d'une collision dans un ensemble de données aléatoires prendrait plusieurs milliers d'années sur une station de travail ordinaire.
Nous considèrerons par la suite que les fonctions de hachage utilisées ne sont pas réversibles facilement.

Exemples de résultats de l'application d'une fonction de hachage :

En utilisant la fonction MD5, qui prend en entrée une suite de bits et produit une empreinte de 128 bits, soit plus de $3{,}4 * 10^{38}$ empreintes possibles.

Soit x = « Texte en clair ».
y = h(x) = « 215eecf4c31fd797adb23b8e2c407ebe ». y est représenté ici en hexadécimal (base 16).

La fonction SHA-1 (Secure Hash Algorithm), plus récente, produit une empreinte de 160 bits, soit plus de $1{,}46 * 10^{48}$ empreintes. Le hash de x par la fonction SHA-1 est :
y = h'(x) = « 9f8f84345c22776978c54348849503feb78438f9 »

Fonctionnement général d'un système OTP

Un système OTP est destiné à l'authentification d'utilisateurs. Son fonctionnement peut être vu comme un dialogue entre un utilisateur et un serveur d'authentification. Cette méthode utilise un nom d'utilisateur, (*login*), et un mot de passe secret associé. Pour être authentifié, l'utilisateur doit entrer son login et prouver qu'il dispose du mot de passe.

Étude des systèmes existants et des problèmes posés :

Il est possible de demander à un utilisateur d'entrer simplement son login et son mot de passe. Les dangers que présente ce type d'authentification sont nombreux :

- Si le mot de passe transite sur un réseau sans être crypté, il pourra être intercepté et réutilisé par une tierce personne (*écoute du réseau*).
- Si un logiciel d'enregistrement des frappes de touches (*keylogger*) est installé sur la machine, il enregistrera la frappe du mot de passe.
- Si une personne se trouve derrière l'utilisateur lorsqu'il tape son mot de passe, elle pourra le mémoriser (*shoulder surfing*).

On le voit, les dangers présentés se ramènent à une condition : connaître le mot de passe utilisé à un instant t permettra de le réutiliser à un instant $t+1$.

Solution proposée par les OTP

Le principe des OTP est de ne jamais authentifier deux fois de suite un utilisateur avec le même mot de passe. Cela ne signifie pas qu'il faudra en changer à chaque ouverture de session, le fonctionnement devant être simple pour l'utilisateur. Une suite de mots de passe est générée à partir d'une phrase secrète, et l'utilisateur de ce système devra prouver qu'il est en mesure de générer la même séquence, sans pour autant fournir le mot de passe.

Initialisation d'une base de mots de passe

À l'installation d'une solution OTP, le système va déterminer un nombre de sessions pouvant être ouvertes avant expiration du mot de passe (N = 100, par exemple), puis générer un ensemble de *hashs* pour chaque utilisateur. Le mot de passe d'un utilisateur lui est fourni, en clair, et le système va le hacher N fois, selon la méthode choisie. À ce jour, l'algorithme le plus courant est MD5; d'autres algorithmes, comme MD4 (plus ancien), ou SHA-1 (plus récent et plus sûr) sont moins utilisés.
Pour créer cette empreinte, on va concaténer le mot de passe (*pass*) en clair à une suite aléatoire de caractères alphanumériques en minuscules (de 1 à 16 caractères, respectant la norme ISO-646). Cette suite de caractères est appelée *seed*, pour graine en anglais. Elle permet de ne pas avoir les mêmes résultats pour deux utilisateurs ayant le même mot de passe.
Puis, on hache une première fois (*pass*+*seed*), et on réduit sur 64 bits l'empreinte générée pour obtenir $pass_0$.

Les N empreintes générées sont les empreintes des hashs successifs de $pass_0$. On construit donc $pass_i$. = MD5($pass_{i-1}$) pour i allant de 1 à N.
Chaque empreinte MD5 est ensuite réduite sur 64 bits par la même fonction que pour $pass_0$, et un checksum sur 2 bits (somme de contrôle destinée à la détection des erreurs) est ajouté à la fin. Chaque $pass_i$ fait donc maintenant 66 bits, soit 6 mots de 11 bits. 11 bits décrivent un ensemble de 2048 éléments, et un dictionnaire de 2048 mots de 1 à 5 lettres est utilisé à cet effet.
À la fin de cette étape d'installation, et pour chaque utilisateur, on disposera donc d'une base de N séquences de 6 mots-clés.

Exemple de mots de passe générés

Nous utiliserons ici un utilisateur nommé *user*, et dont le mot de passe est « password », nous utiliserons MD5, le nombre de mots de passe générés pour cet utilisateur est N = 5, et la graine (*seed*) tirée aléatoirement est « abcd123 ».
À l'aide du programme opiekey sous Linux, on peut générer des mots de passe jetables.
À l'exécution, il faut fournir en paramètres de ligne de commande : l'algorithme utilisé, le nombre N, et le nombre de mots de passe à générer en partant de N (en descendant)

```
$ opiekey -5 -n 5 5 abcd123
1: EVE DUE KERN FORK CAW CLAY
2: LEAN LOG MEL BATE EDEN ALSO
3: LINT WEE LEST SAG TEST POD
4: COT LEER OH BLUE CULT BLUR
5: DIME BRAY ACTA VEIN WEED RAID
```

Authentification d'un utilisateur par OTP

L'utilisateur veut s'authentifier après la mise en place du système. Dans la base de données, il reste 5 mots de passe correspondant à son login. Comme on l'a vu, l'authentification se fait par l'apport de la preuve de la détention du mot de passe, et non par sa communication directe. Les serveurs d'authentification va donc adresser un *challenge* à l'utilisateur. Ce challenge est une chaîne de caractères, composée de trois séquences séparées par des espaces.

- La première contient le nom de l'algorithme. Par exemple : otp-md5.
- La seconde est le numéro du mot de passe que l'on attend de l'utilisateur (numéro de séquence).
- La troisième est la graine (*seed*).

Cette demande est transmise en clair à l'utilisateur, qui reçoit ici : otp-md5 5 abcd123.

De son côté, il calcule, ou a déjà calculé la phrase d'authentification, et la renvoie au serveur. À la prochaine authentification, le serveur demandera le mot de passe numéro 4, puis 3, jusqu'à épuisement de la base de hashs.

Il fournira donc « `DIME BRAY ACTA VEIN WEED RAID` » à la première ouverture de session, puis lors de la session suivante « `COT LEER OH BLUE CULT BLUR` », puis « `LINT WEE LEST SAG TEST POD` », etc.

Pourquoi ce système est-il sécurisé ?

Nous avons vu qu'un attaquant pouvait espionner le réseau afin de lire les mots de passe transitant du client au serveur. S'il arrive à obtenir le mot de passe N, il ne pourra le réutiliser pour la session suivante, qui demandera le mot de passe N-1. Il ne pourra pas non plus le déduire, puisque passN est créé en hachant passN-1, et l'on a vu qu'il n'était pas envisageable d'inverser une fonction de hachage.

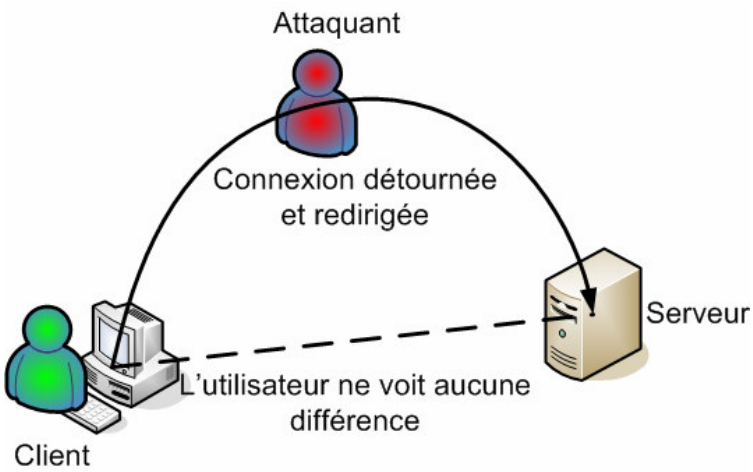
Le serveur d'authentification est également sécurisé : il ne stocke aucun mot de passe en clair, empêchant ainsi un intrus d'avoir accès au mot de passe lui-même.

De plus, le temps de calcul des empreintes numériques rend moins courante les attaques par recherche exhaustive.

Les limitations des systèmes OTP

Un tel système d'authentification permet seulement de protéger son mot de passe contre une éventuelle réutilisation. Trois contreparties sont à considérer avant d'adopter un tel système :

1. Le challenge proposé ne varie pas, et le nombre de hachages à effectuer est donné à tout le monde. Cela signifie qu'un attaquant pourrait tenter une attaque par force brute (recherche exhaustive du mot de passe), ou utiliser un dictionnaire de mots pour casser le mot de passe. Cette recherche est rendue difficile par le temps de calcul nécessaire à la fabrication d'une réponse à un challenge, mais une ou plusieurs machines peuvent casser un mot de passe simple (comme « *password* » par exemple). Une limitation du nombre d'accès peut contrer ce type d'attaque.
2. Le système révèle des informations sur l'existence et les connexions des utilisateurs. En effet, si un attaquant essaye d'ouvrir une session deux fois de suite avec un login existant, le nombre de hashs demandés lors de l'envoi du challenge sera décrémenté. Si le compte n'existe pas, le système d'authentification enverra deux fois de suite des challenges de numéro de séquence non consécutifs, avec des graines différentes. De plus, la vérification périodique de la variation du numéro de séquence permet de savoir, si deux numéros ne sont pas consécutifs pour deux vérifications, qu'un utilisateur a fait une demande d'ouverture de session.
3. Les OTP ne protègent pas contre une attaque dite « Man in the middle » (MITM). Ce type d'attaque consiste à intercepter le trafic réseau et à se faire passer pour une machine, ici le serveur OTP. Un utilisateur qui voudrait se connecter demande un challenge (à la mauvaise machine), qui relaie la demande auprès du vrai serveur d'authentification, et se place entre le client et le serveur pour toute la durée de la communication, récupérant tout le flux de données au passage.



Attaque de type Man-in-the-middle

Ce type d'attaques est difficile à détecter car la machine qui relaie le trafic agit de manière transparente, sans modifier les flux. Cette machine ne pourra pas connaître le mot de passe en clair, mais pourra surveiller les actions effectuées, puis utiliser la session après déconnexion du client.

Nous étudions ici le login et la déconnexion de user, à l'adresse 10.0.0.5, sur la machine 10.0.0.2 (le serveur d'authentification). La machine 10.0.0.4 intercepte tout le trafic du réseau, et retransmet de manière transparente.
Voici un exemple d'usurpation de session : si un observateur extérieur surveillait le réseau, voici ce qu'il verrait :

Connexion de 10.0.0.5 (client) vers 10.0.0.2 (serveur)

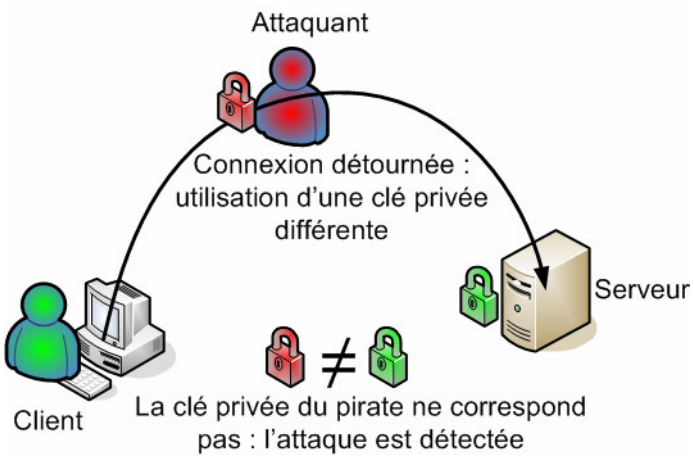
Envoyé par le client	***Envoyé par le serveur***	***Relayé***
	`Linux 2.6.10 (10.0.0.2)` `login:`	oui
`user`		oui
	`otp-md5 5 abcd123,` `Response:`	oui
`EVE DUE KERN FORK CAW CLAY`		oui
	`Welcome, user` `bash $`	oui
(Le client utilise sa session, tout est relayé et il ne s'aperçoit de rien.)		
`logout` `(Client déconnecté)`		non
À ce moment, le client se déconnecte, mais la machine espionne ne relaie pas cette information et conserve la session de l'utilisateur afin d'accéder à ses documents.		

Ce type d'attaque est très facile à mettre en oeuvre, et difficilement détectable.

Combinaison avec SSH

Pour profiter des avantages des OTP en évitant ce type d'attaques, il est préférable d'associer une authentification par OTP combinée à une connexion sécurisée (SSH).
Le serveur SSH possède une clé unique, privée, que personne ne connait et qui sert au décryptage des communications sécurisées. À cette clé privée est associée une clé publique, utilisée pour le cryptage. Une analogie résume bien le principe : la clé publique correspond à un coffre-fort ouvert, dans lequel on place des documents et que l'on referme sans clé, et que l'on envoie au destinataire qui est le seul à posséder la clé privée permettant de l'ouvrir. De cette manière, il est possible au client de vérifier que la personne à qui il envoie ses données connaît la clé privée du serveur, donc qu'elle est le serveur (La clé privée n'est pas partagée entre plusieurs utilisateurs et doit rester secrète).

Exemple d'un détournement de session SSH :



Dans le cas d'une attaque par détournement de connexion SSH, voici ce qui apparaît sur l'écran du client lors de l'authentification :

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@       WARNING: HOST IDENTIFICATION HAS CHANGED!         @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
```

Traduction : *Attention : L'identification de l'hôte a changé ! Il est possible que quelqu'un soit en train de faire de vilaines choses ! Quelqu'un pourrait être en train de vous surveiller en ce moment (attaque MITM).*

Voici les avantages d'une telle combinaison : d'une part, le client n'a pas tapé son mot de passe, et a ainsi évité qu'un logiciel espion ne le retienne. Si un logiciel espion a enregistré ses frappes de touche, il ne pourra réutiliser le mot de passe donné, qui ne sera plus valable à la prochaine ouverture de session. D'autre part, il a détecté qu'un détournement était en cours, et est capable d'authentifier la machine avec laquelle il communique.
Cette combinaison assure un bon niveau de sécurité pour les utilisateurs du système d'information.

Ajout d'une sécurité supplémentaire : les tokens OTP

Pour qu'un utilisateur évite de taper sa phrase d'authentification OTP, il est possible de le munir d'un *token OTP*, clé physique générant des mots de passe à usage unique après entrée d'un code PIN (*Personnal Identification Number*) par l'utilisateur. Ces clés permettent d'avoir à se passer de clavier, notamment pour l'authentification des utilisateurs dans un contexte différent du terminal informatique (accès physique à des locaux, activation de machines).

La société RSA Security, renommée dans le domaine de la cryptographie et de l'authentification, commercialise la solution SecureID, au prix de 3500 euros pour 25 utilisateurs et pour 3 ans.

Références

- RFC 1760, The S/KEY One-Time Password System
- RFC 2289, A One-Time Password System
- MISC Magazine n°15 – De l'usage des mots de passe jetables
- One-Time Passwords in Everything (OPIE) : http://inner.net/opie/
- Fonctions de hachage : http://fr.wikipedia.org/wiki/Fonction_de_hachage
- SSH : http://fr.wikipedia.org/wiki/SSH
- SecureID : http://www.rsasecurity.com/node.asp?id=1156