TD8 – Sécurité et protocoles cryptographiques (1ère partie)

{Emmanuel.Godard, Yann.Esposito}@lif.univ-mrs.fr

24 novembre 2005

☞ Le but de ce TD va être de se familiarisé avec les systèmes de sécurité.

1 Protection standard

1.1. Rappelez comment protéger un réseau avec un pare-feu

Il suffit de mettre un pare-feu entre l'extérieur et l'intérieur du réseau local

(a) Quelle est l'efficacité d'un tel système, à quelles attaques est-ce sensible ?

Le pare-feu est un système très restreint. Il permet simplement de couper certaines communications. En particulier, il permet d'éviter de laisser passer des port qui ne sont pas autorisés.
Mais les défenses d'un pare-feu peuvent aisément être contournées en utilisant des tunnels par exemple.
En tout cas, il s'agit de la première mesure minimale à mettre en œuvre pour protéger un réseau.

1.2. Qu'est-ce qu'un pot de miel ?

Il s'agit d'une machine rendue volontairement vulnérable à des attaques afin d'attirer des pirates et d'étudier leur stratégies d'attaque pour mieux les comprendre et les anticiper.

(a) Quels sont les risques liés à l'installation d'un pot de miel ?

Le pot de miel se faisant attaquer par un pirate. Celui-ci pourra alors utiliser ce pot de miel pour attaquer le reste du réseau local... En particulier, il pourra le scanner ou ou directement attaquer le reste de l'infrastructure.

(b) Quelle configuration devrait-on alors adopter ?

Pour éviter qu'un pirate puisse attaquer le réseau local, il faut **contrôller les données**. De plus l'intérêt du pot de miel va être de **capturer les données**.
Pour que le pot de miel soit utile, il faut que le pirate piégé se sente à l'aise. Il doit pouvoir effectuer des connexions extérieures, récupérer ses rootkit, joindre des canaux irc...
Il faut donc que le pirate puisse communiquer avec l'extérieur mais ne pas nuire à l'extérieur. Pour cela, on peut bloquer les connexions après un certain volume ou un certain temps. Dans les pots de miel récents, on pourra même modifier directement ces communications afin de les rendre inffensives.

La capture des données étant le but du pot de miel, il faudra écrire les logs dans un endroit sûr que le pirate ne pourra pas compromettre et à son insu. Le plus sûr étant sur une sortie imprimante...

2 Attaque par recherche exhaustive (brute force)

☞ En 1999, un groupe de cryptographiste on construit un craqueur de DES. Il était capable de tester les 2^{56} clés de DES en 56 heures. La machine coûtait 250 k$.

2.1. En extrapolant la loi de Moore. Combien de temps faudrait-il aujourd'hui à une machine similaire pour casser une clé DES ?

La loi de Moore stipule que la vitesse des ordinateur double tous les 18 mois. Cela fait six ans que la machine a été créée. Ainsi une machine similaire serait aujourd'hui $2^{72/18} = 2^4 = 16$ fois plus rapide.
Il faudrait donc : $\frac{56}{16} = 3,5$ heures pour casser une clé DES.

☞ MD5 est une fonction de hachage. Récemment, on a pu montrer que l'on pouvait calculer des collision. Une chercheuse déclarait même qu'elle arrivait à trouver des collision à la main.
Aussi, d'autres fonctions de hachages existent ; nottamment SHA-1. SHA-1 produit des "hash" de 160 bits. La probabilité de trouver une collision de SHA-1 au hasard est de 1 sur 2^{80} exactement. Si vous hachez 2^{80} messages, vous êtes sûr de trouver au moins une paire ayant le même "hash".

2.2. Combien de temps faudrait-il théoriquement pour trouver une collision de SHA-1 en utilisant une machine capable de générer 2^{69} "hash" en 56 heures. Sachant que l'on pourrait construire aujourd'hui un tel ordinateur qui coûterait entre 25 et 38 millions de dollars ?

Il faudrait $2^{80-69} * 56$ heures, c'est-à-dire $2048 \times 56 = 114688$ heures ou 4778 jours ou encore $13 ans$.

2.3. Vous semble-t-il raisonnable d'utiliser SHA-1 comme fonction de hachage aujourd'hui ? Pour combien de temps ?

En réalité, une équipe de chercheur à trouver une manière de faire des collision qui ne nécessite plus 2^{80} essais mais seulement 2^{69}. Ainsi ce résultat théorique rend caduque l'utilisation de SHA-1 comme fonction de hashage correcte.
De plus il existe un algorithme qui permet de retrouver un autre message ayant exactement le même hash qu'un hash donné en ne regardant que 2^{106} messages ; c'est-à-dire beaucoup moins que les 2^{160} espérés.

3 Protocoles

Commentez et critiquez les protocoles suivants (description, vulnérabilités, contexte d'utilisation)

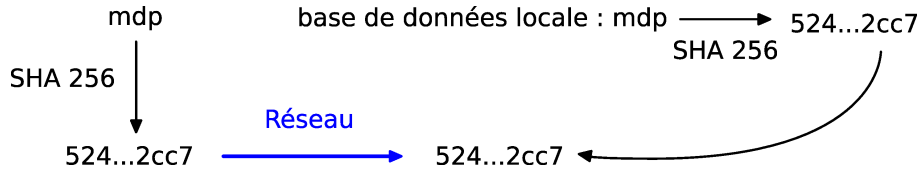
3.1. Soit la méthode de chiffrement simple et robuste suivante :
- C_0 est la clé de session
- $C_i = M_i \oplus C_{i-1}$

Les messages passant sur le reseau seront :
$M'_1 = M_1 \oplus C_0$
$M_2 \oplus M'_1$
...
Il suffit donc de tout écouter et pour traduire de refaire des $\oplus$ dans le bon ordre. Cette méthode de chiffrement n'est donc absolument pas robuste...

3.2.

Ce n'est pas réellement une méthode de chiffrement. Il faut savoir que c'est exactement le même genre de transformations (si on peut appeler l'identité une transformation) qui sont appliqués dans les protocoles POP, IMAP et SMTP.

3.3.



C'est robuste, mais on ne peut pas utiliser ce système pour communiquer de longs messages. Sinon, les même hash finiraient par revenir de plus en plus souvent et le système deviendrait attaquable facilement avec des methodes statistiques.
Ce protocole est proche de celui utilisé pour la gestion des mots de passe sous la plupart des systèmes UNIX.

3.4.

$$A \to S : r_{Ak}$$
$$S \to B : Aveutcommuniquer$$
$$B \to S : r_{Bk}$$
$$S \to A : r_B \oplus r_A$$

Attaque utilisant : $x \oplus x \oplus y = y$
X et Y sont des attaquants.

$$X \to S : r_{Xk} \times r_{Ak} = r_X \times r_{Ak}$$
$$S \to Y : Xveutcommuniquer$$
$$Y \to S : r_{Yk}$$
$$S \to X : r_Y \oplus r_X \times r_A$$

Connaissant r_X et r_Y on peut en déduire r_A.