Cryptographie

notes Cryptographie

Table des matières

1. SSH

Système de crytographie à clefs publiques permettant de remplacer les traditionels rlogin, rsh, rcp par leur équivalents sécurisé (ssh, scp, sftp), et de mettre en place des canaux de communication criptés entre 2 machines. Le daemon sshd doit être lancé sur la machine cible pour accepter les connexion. Les clefs publiques et privés ainsi que les listes d'accès authorisés (hôte et utilisateurs) sont placées traditionnellement dans le répertoire ~/.ssh de l'utilisateur

Pour éviter les ataques par usurpations d'adresses IP, chaque client stoque les clefs publiques des hôtes sur lequels il se connecte à la première connexion dans les fichiers known_hosts pour SSHv1 et know_hosts2 pour SSHv2. Aux connexions suivantes, le client envoie un chalenge à l'hôte, chiffré avec sa clef publique, pour vérifier qu'il est bien en possession de la clef privée correspondante. En cas d'echec, le client refuse de se connecter et envoie un message d'alerte.

1.1. Gestion des clefs

- génerer des clefs
  ```
  >ssh-keygen -t (rsa1|rsa|dsa)
  ```
  - pour l'hôte : ne pas mettre un mot de passe vide pour la clef privée
  - pour chaque utilisateur: protéger la clef privée par mot de passe. Génère les clefs publiques (identity.pub,id_rsa.pub,id_dsa.pub) et privées (identity,id_rsa,id_dsa)
- changer le mot de passe associé à une clef privée
  ```
  >ssh-keygen -p -t (rsa1|rsa|dsa)
  ```

1.2. Connexion

1. ```
   >ssh -l utilisateur hôte
   ```
   le client vérifie que la clef privé de l'hôte n'a pas changé en lui envoyant un chalenge et envoie la clef publique de l'utilisateur local à l'hôte
2. L'hôte teste la présence de la clef publique dans ses listes d'accès (~/.ssh/authorized_keys et ~/.ssh/authorize_keys2) de l'utilisateur visé, et envoie, dans l'affirmative, un chalenge au client pour vérifier qu'il dispose bien de la clef privée.

2. GPG (GnuPG)

Système de cryptographie à clefs publiques pour le (de)cryptage de documents confidentiels

- générer des clefs
  ```
  >gpg --gen-key
  ```
- générer une clef de revoquation
  ```
  >gpg --gen-revoke
  ```
  ```

- afficher le porte clef
```
>gpg --list-keys
```
- modifier une clef perso
```
>gpg --edit-key
```
- Sortir la clef sous un format transportable par email
```
>gpg --armor --export clef_id
```
- Importer une clef
```
>wget $url -O - | gpg --import
```

3. Systèmes de fichiers

3.1. sur périphérique de bouclage

Un périphérique de bouclage permet de représenter un système de fichier comme étant un fichier pour le manipuler, de même que les périphériques sont en général manipulables sous la forme de fichiers "bloc" placés dans /dev.

1. création d'un fichier réceptacle de 10Mo
```
>dd if=/dev/zero of=image bs=1M count=10
```
2. création d'un système de fichier ext2 par association du fichier créé avec un périphérique de bouclage
```
>losetup /dev/loop0 image
>mkfs -t ext2 /dev/loop0
>losetup -d /dev/loop0
```
ou directement dans le fichier
```
>mkfs -t ext2 image
```
3. montage/démontage
```
>losetup /dev/loop0 cd.img
>mount -t ext2 /dev/loop0 $point_montage
>umount $point_montage
>losetup -d /dev/loop0
```
ou laisser à la commande mount le soin d'allouer un périphériphérique /dev/loop?
```
>mount -t ext2 -o loop image $point_montage
>umount $point_montage
```

3.2. cryptés

On peut crypter n'importe quel système de fichiers en utilisant les périphériques de bouclage.

- pour la création du fichier receptacle: utiliser /dev/urandom plutôt que /dev/zero pour qu'on ne puisse pas savoir quelles parties sont occupées et lesquelles ne le sont pas.
```
>dd if=/dev/urandom
                              of=image bs=1M count=10
```
- l'option -e de losetup et -o encryption= permettent de préciser l'algorithme de cryptage.
```
>losetup -e aes /dev/loop0 cryptfile
>mount -t ext2 -o loop,encryption=aes image
```

a partir d'un périphérique (lecteur de carte flash ou partition disque dur) la démarche est la même sauf qu'on ne précise pas la taille à la création, et que le fichier receptable est en fait le fichier périphérique associé dans /dev.

3.3. compréssés

plusieurs approches existent, pour économiser de la place au détriment en général de la souplesse d'utilisation

- utiliser un système de fichier en lecture/écriture qui supporte la compression transparente comme ext2compr. le problème principal étant la récupération en cas de corruption puisque une erreur quelque part dans la table des inodes, empêche la lecture de toute la table.
- utiliser un système de fichier compréssé en lecture seule comme squashfs, cramfs, ... Il faut au préalable au montage, créer le système de fichier.
- utiliser un périphérique de blocs compréssés comme compress_loop du projet knoppix. Il faut également au préalable créer le système de fichier. l'avantage étant qu'on peut utiliser virtuellement n'importe quel système de fichier, et qu'on peut faire une combinaison compression+encryption sur ce système de fichier

3.3.1. squashfs

squashfs est un système de fichier compréssé en lecture seule. L'installation passe par un correctif et une recompilation du noyau.

- création
  ```
  >mksquashfs $repertoires
  ```
  si l'image existe déjà elle est augmenté des nouveaux fichiers.
- montage
  ```
  >mount -t squashfs image $point_de_montage
  ```

3.3.2. compress_loop

compress_loop permet d'utiliser des périphérique de bouclages compréssés.

1. création
   1. création d'une image d'un système de fichier $type sur périphérique de bouclage standard par une méthode quelconque. Le plus pratique étant un système iso9660, puisque mkisofs permet la création en vol, sans a avoir a prévoir une taille pour le fichier receptacle.
   2. compression de l'image
      ```
      >create_compressed_fs image 65536 > image.z
      ```
2. montage

```
>modprobe compressed_loop
>losetup /dev/cloop/0 image.z
>mount -t $type /dev/cloop/0 $point_de_montage
```

3. démontage

```
>umount $point_de_montage
>losetup -d /dev/cloop0
```

3.4. compréssés et cryptés

On peut utiliser:

- un système de fichier compréssé (squashfs) sur périphérique de bouclage crypté, ou
- n'importe quel système de fichiers sur périphérique de bouclage compréssé/crypté

3.4.1. squashfs

C'est la plus simple des méthodes, et elle ne diffère pas du cryptage d'un système fichiers quelconque

3.4.2. compress_loop

Pour compresser/crypter un système de fichier quelconque, on doit passer par 2 périphériques de bouclage en l'absence d'un périphérique spécifique qui réaliserai cette opération en une seule passe. L'ordre étant compression d'abord, chiffrement ensuite, puisque le chiffrement génére beaucoup de bruit qui limite l'éfficacité de la compression.

1. création
    1. création d'une image standard
    2. création d'un fichier receptacle crypté image.z sur /dev/loop0, de taille 3x inférieure à la taille de l'image standard
    3. compression de l'image standard vers /dev/loop0

    ```
    >create_compressed_fs image 65536 > /dev/loop0
    ```
2. montage
    1. création d'un périphérique de bouclage crypté sur l'image

    ```
    >losetup -e aes /dev/loop0 image.z
    ```
    2. création d'un périphérique compréssée à partir de /dev/loop0

    ```
    >losetup /dev/cloop/0 /dev/loop0
    ```
    3. montage normal

3.5. montage à la connexion

4. SSL/TLS

4.1. Cryptage/Décryptage

- crypter avec un système symétrique
```
>openssl enc -aes256 -in $file -pass fd:0 -out ${file}.cpt
```
- decrypter avec un système symétrique
```
>openssl enc -d -aes256 -in $file -pass fd:0
```

4.2. Authorité de certification

Pour mettre en place un réseau de confiance basée sur une infrastructure à clefs publique (PKI) et SSL, il faut

- acheter un certificat auprés d'une autoritée connue sur le net (netscape, thawte, verysign, ...), ou
- mettre en place sa propre autorité de certification (CA)

Mettre en place sa propre CA

- Générer l'arborescence de la CA
```
>cd /var
>/usr/lib/ssl/misc/CA.pl -newca
```
- Générer et signer (par le CA) des clefs/certificats sans mots de passe (pour serveurs/services)
```
>openssl req -new -nodes -keyout newreq.pem -out newreq.pem
>/usr/lib/ssl/misc/CA.pl -sign
>mv newcert.pem server_crt.pem ;  mv newreq.pem server_key.pem
```
- Générer et signer des clefs/certificats avec mots de passe (pour utilsateurs)
```
>openssl req -new -keyout newreq.pem -out newreq.pem
>/usr/lib/ssl/misc/CA.pl -sign
>mv newcert.pem server_crt.pem ;  mv newreq.pem server_key.pem
```

Installer les clefs

l'installation des clefs/certificats se fait dans `/etc/ssl/certs`. Les clefs ne doivent être lisible que par les services qui les envoient (mode 600), et les certificats doivent être lisible par tout le monde (mode 666)

1. créer un répertoire dans `/etc/ssl`
2. lier les servicats/clefs de ces services de `/etc/ssl/certs` dans ce répertoire,
3. configurer le service pour qu'il aille chercher les clefs/certificats au bon endroits

5. PAM

- Durcir les mots de passes par des empreintes de mots de passes blowfish éditer /etc/pam.d/system-auth
```
|auth        sufficient    /lib/security/pam_unix.so likeauth bigcrypt
nullok
```
- Monter automatiquement des systèmes de fichiers crypté au démarage (module pam_mount)

pam_mount permet de monter un système de fichiers à la connexion d'un utilisateur. C'est un module pam, ce qui rend l'autentification très souple en cas de système de fichiers crypté. En général les systèmes de cryptographie symétriques impose une phrase de passe qui en général beaucoup plus longue que le mot de passe de connexion. Pour n'avoir a taper qu'un mot de passe à la connexion, pam-mount permet d'envoyer la phrase de passe à notre place:

1. crypter la phrase de passe avec le mot de passe de connexion et de placer le résultat dans un fichier (/etc/ehb/burghard) avec l'algo bf
```
>echo $pass | openssl enc -bf -pass fd:0 >/etc/ehb/burghard
```
2. éditer /etc/security/pam-mount.conf
```
|volume burghard local - /dev/sda1 - - bf /etc/ehb/burghard
```
3. éditer /etc/fstab pour donner les caractéristiques du système a monter sur /dev/sda1
```
|/dev/sda1 /home/burghard/Data ext3
user,loop,encryption=aes256,noauto,noatime 0 0
```
4. éditer un /etc/pam.d/kde3 pour déclencher le montage à la connexion dans kde avec le mot de passe de connexion
```
|auth       optional     /lib/security/pam_mount.so
|auth       required     /lib/security/pam_stack.so
service=system-auth try_first_pass
|...
|session    optional     /lib/security/pam_mount.so
```

- Ajouter automatiquement les clefs publiques dans l'agent ssh en utilisant le mot de passe de connexion
  1. modifier /etc/pam.d/kde3
```
|auth       optional     /lib/security/pam_ssh.so
|auth       required     /lib/security/pam_stack.so
service=system-auth try_first_pass
|...
|session    optional     /lib/security/pam_ssh.so
```

6. SASL

7. XAUTH