

TP de sécurité Unix et réseau

Ronan KERYELL
Alain LEROY
Pascale MÉNARD

—
Laboratoire Informatique & Télécommunications
Département Informatique
École Nationale Supérieure des Télécommunications de Bretagne

—
3A IT-ISIC

2 février 2003

Résumé

Ce TP propose quelques tâches liées à l'administration sécurisée de machines Unix et de réseaux IP.

Avertissement : ce TP vous prépare à être éventuellement administrateur système et réseau dans votre vie professionnelle future. Il est clair que les techniques exposées ici sont à utiliser selon l'éthique de cette profession et non dans des buts illicites.

1 Introduction

Le réseau que nous allons utiliser a la structure indiquée en figure 1 qui permet de simuler un réseau d'entreprise avec un filtrage de certains paquets IP.

On commencera par vérifier le câblage du réseau au cas où les encadrants se seraient trompés.

Pour configurer les machines, passer sous `root` et

```
cd TPsec  
./tpconf
```

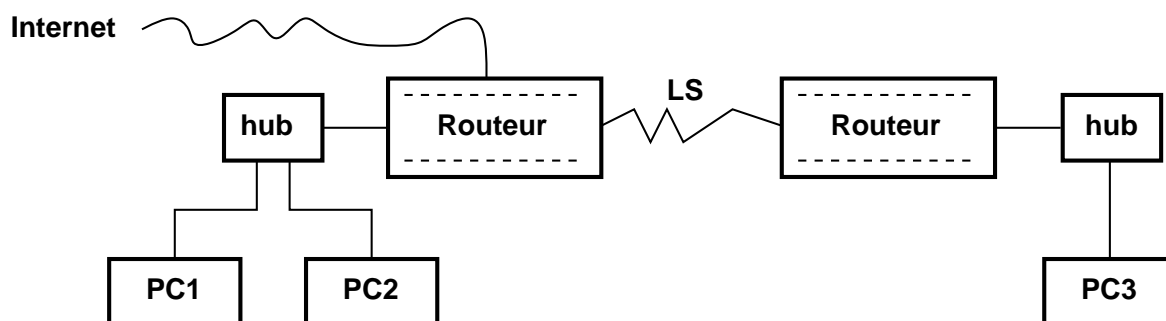


FIG. 1 – Architecture du réseau expérimental

Pour configurer les routeurs, se connecter via `minicom` et configurer par le merveilleux protocole de transport assisté par souris à partir du fichier `configrouteur` du répertoire précédent.

La connexion à Internet devrait marcher grâce à de la translation d'adresse.

L'année prochaine on essaiera de faire plus compliqué avec une DMZ ☺.

2 Test de robustesse des mots de passe

Avant que ce ne soit fait par un pirate, il est indispensable de tester si un utilisateur n'a pas un mot de passe trop facile par rapport à une attaque par dictionnaire.

Pour ce faire il convient à un administrateur de faire tourner régulièrement un outil du style `crack`, `L0phtCrack` ou `John the Ripper` et de bloquer les comptes qui ont été cassés. Ces comptes ne seront ré-activés qu'après changement de mot de passe sérieux.

L'intérêt de `John the Ripper` est qu'il peut être rajouté directement au système de PAM du système (`pam_passwdqc`) afin de refuser à un utilisateur la possibilité de mettre un mot de passe craquable facilement par n'importe qui. C'est aussi un logiciel portable, optimisé et pouvant casser des mots de passe Windows, etc.

2.1 Installation de John the Ripper

Pour des raisons de performance on va installer ce logiciel sur les machines `pc3ainfox` et non sur les machines de la salle de TP.

Récupérer sur <http://www.openwall.com/john/> une version récente.

Décompresser avec un `tar zxvf` et lire le `README`.

Compiler suivant le mode d'emploi¹.

2.2 Récupération du fichier de mots de passe chiffré

On peut s'entraîner avec comme source les mots de passes chiffrés de l'École qu'on peut récupérer sur une machine élève par un

```
ypcat passwd > passwd.1
```

car les NIS sont utilisés pour distribuer des tables système de manière globale à l'École. Il est clair que dans la vraie vie il ne faudrait pas utiliser les NIS mais utiliser un système qui rend la vie plus difficile aux pirates du dimanche...

Rapatrier le fichier `passwd.1` dans le répertoire `run` du logiciel.

2.3 Cassage des mots de passe

Dans le répertoire `run` lancer déjà avec la configuration de base

```
./john passwd.1
```

et constater avec horreur qu'il y a déjà beaucoup de mots de passe trouvés ☺. Néanmoins cela prend du temps alors passer à la suite pendant que l'ordinateur travaille.

¹Cela permet au passage de se familiariser avec les techniques classiques de compilation de logiciels et d'installation à partir des sources...

Réfléchir aux méthodes de communication positives permettant d'éviter que de pareils cas se reproduisent. Évidemment cela n'empêche pas de faire tourner régulièrement un logiciel de craquage afin d'éviter ce genre d'étourderies. La bonne nouvelle est que ce logiciel possède un mode qui envoie un courriel à une personne dont le mot de passe est trouvé afin de la prévenir de la faiblesse trouvée.

Les plus curieux regarderont le fichier de configuration `john.ini` en conjonction avec le fichier explicatif `doc/RULES`. Par défaut le système utilise les noms des utilisateurs comme source de mot de passe, puis des combinaisons du petit dictionnaire `password.lst` avant de se lancer dans une recherche par force brute.

Pour une mise en production dans la vraie vie il faudrait utiliser un dictionnaire de mots plus important.

2.4 Conclusion

À retenir :

- ne pas utiliser les NIS pour partager les mots de passes car cela fait passer sur le réseau les mots de passe hachés sur le réseau ;
- ne plus utiliser les hachages classiques d'Unix sur 56 bits car c'est de la technologie vieille de 30 ans mais plutôt des hachages sur au moins 128 bits ;
- utiliser régulièrement des outils de craquage de mots passe pour éliminer les mots de passe trop faibles des utilisateurs et intégrer ces outils aux systèmes de changement de mots de passe.

3 Étude de protocoles non sécurisés

3.1 Concepts

Afin de se convaincre d'utiliser des protocoles sécurisés nous allons observer en quoi consistent les protocoles « de base » existant sur Internet depuis l'origine des temps.

Dans un premier temps on peut lancer un programme `tcpdump` ou `ethereal` et faire un `telnet` ou un `ftp` entre 2 machines. Que constatez vous ?

Suggestion d'un élève d'un TP précédent : demander à votre binôme de commander quelque chose sur son site Internet préféré avec sa carte bancaire et son code secret et regardez pendant ce temps-là ce qui passe sur le réseau pour vérifier le niveau de sécurité. ☺

3.2 Utilisation de `dsniff`

On va automatiser le concept en utilisant le logiciel `dsniff` qui est capable de reconstruire le contenu d'une communication dans de nombreux protocoles.

Lancer sur le PC2 un

```
dsniff -m
```

et essayer de vous connecter par exemple du PC3 vers le PC1 avec un protocole non sécurisé style `telnet` ou `ftp` (ne tapez pas votre vrai mot de passe ☺). Que constatez-vous à la fin de la connexion ?

3.3 Faire du faux courrier électronique ?

On pourra aussi regarder le courrier électronique (protocole SMTP). Le problème est que la charte d'usage du réseau à l'ENST Bretagne stipule que c'est interdit *même si c'est à la demande d'un enseignant* et qu'actuellement on n'a pas de courrier fonctionnant en interne dans le bac à sable du TP...

Constatez que le protocole est très simple comme l'indique le S du nom.

Il est donc très simple d'envoyer du courrier électronique avec un simple

```
telnet un-serveur-de-mail mail
```

et de causer directement en SMTP².

Est-ce gênant ? Oui et non : le courrier standard n'offre pas plus de garanties et on peut faire de fausses lettres ou des lettres anonymes encore plus facilement en postant discrètement dans une boîte aux lettres.

Il existe des protocoles d'échange plus sécurisés entre les facteurs et on peut de surcroît envoyer des lettres chiffrées au niveau du contenu indépendamment de l'enveloppe et de son transport (voir § 8).

Sinon pour le luxe du pirate, `mailsnarf` reconstruit au format courriel UNIX du courriel qui a été capturé par `dsniff` et qui peut donc être lu par son outil favori ensuite...

3.4 Conclusion

Mettre à la poubelle les protocoles non sécurisés `telnet`, `FTP`, `rsh`,... et utiliser à la place `ssh`, `scp`, `sftp`,...

Imaginez les attaques faisables sur un réseau sans fil non sécurisé avec de l'espionnage de paquets ou des attaques par interception au milieu : sécurisez la configuration réseau et utilisez de toute manière des protocoles sécurisés.

Ne pas croire tous les courriels que vous recevez, même de personnes connues.

4 Filtrage IP

La section précédente a montré clairement les limites des protocoles simples mais non sécurisés. Une première limitation va être d'en limiter l'usage depuis l'extérieur par exemple pour 2 raisons :

- si quelqu'un même de bien intentionné les utilise il peut être espionné et quelqu'un pourra se connecter à sa place ;
- autant limiter au strict minimum les connexions extérieures afin de limiter un risque lié à l'exploitation d'un trou de sécurité dans un protocole inutilisé.

Le filtrage concernera sur la figure 1 la machine PC3.

4.1 Outil d'analyse de ports ouverts `nmap`

Attention : certains administrateurs systèmes considèrent que le simple fait qu'on regarde quels sont les ports ouverts constitue une violation...

Essayer cet outil pour tester les ports IP ouverts sur diverses machines. Comparer par exemple `www.lit.enstb.org`, `www.enst-bretagne.fr`, `cig.ensmp.fr`, `cri.ensmp.fr`, `www.microsoft.com`.

²HELP permet d'avoir le mode d'emploi en ligne.

Il existe une version X11 `xnmap` pour les amateurs de cliquodrome.
Éventuellement lancer un outil d'espionnage style `tcpdump` ou `ethereal` en même temps pour comprendre comment `nmap` peut fonctionner aussi rapidement malgré les temporisations protocolaires (TCP,...).

4.2 Filtrage avec syntaxe à la CISCO

Cette partie nécessite un routeur CISCO ou un logiciel à syntaxe compatible tel que Zebra.

4.2.1 Syntaxe CISCO de base

Voir la documentation du constructeur, le cours de sécurité et les TP précédents. Pensez à l'utilisation de `<TAB>` pour la complétion automatique et `?` pour l'affichage de l'aide contextuelle.

Pour passer en mode super-utilisateur sur le routeur :

```
enable
```

Pour afficher la configuration actuelle :

```
show running-config
```

et la configuration plus précise des listes de contrôle d'accès avec l'utilisation de chaque règle :

```
show access-lists
```

Pour entrer dans le mode de configuration interactive :

```
configure terminal
```

puis pour en sortir et valider la configuration tapée :

```
exit
```

Pour configurer l'`access-list` 100 par exemple :

```
access-list 100 contenu d'une ligne l'access-list
```

Pour entrer dans la configuration d'une interface :


```
interface le-nom-de-l'interface
```

pour associer par exemple la liste contrôle d'accès 100 à cette interface :

```
ip access-group 100 in ou out
```

et en sortir :

```
exit
```

 Une fois rajouté une `access-list` a une interface, tout le trafic est bloqué tant qu'on ne l'a pas explicitement autorisé avec des règles de filtrage.

4.2.2 Filtrage à effectuer

Autoriser en sortie le trafic `www` seulement vers le mandataire `www` de l'école afin d'en obliger l'usage pour mieux exploiter le cache global. Un tel mandataire global pourrait aussi servir à filtrer du contenu³.

On autorisera toutes les connexions sortantes pour ne pas emprisonner les utilisateurs⁴.

En entrée on laissera le port `ssh`, le courrier électronique, le `www` et le `DNS` (`domain`). On interdira spécifiquement les `telnet`, `rsh` et autres `rlogin` ou `ftp`.

Pour s'inspirer sur les noms des services et leur numéros, regarder dans `/etc/services` ou bien comme d'habitude chercher dans le `www`.

Dans la suite on fera des expérimentations avec `nmap` pour vérifier depuis l'intérieur ou l'extérieur ce qu'on peut faire réellement.

Remarquez au passage l'intérêt d'avoir un compte `root` à l'extérieur ne serait-ce qu'un accès par modem à un fournisseur d'accès Internet pour faire de l'audit du réseau avec des outils comme `nmap`, etc.

4.2.3 Faire une configuration libertaire

Dans cette approche on n'interdit que le trafic potentiellement dangereux ou non sécurisé. C'est bien pour les utilisateurs mais dangereux car le moindre trou de sécurité à la mode a le plus de chances d'être exploité.

4.2.4 Faire une configuration paranoïaque

On commence par tout interdire.

Seuls les services indispensables seront ouverts.

4.3 Utilisation d'`iptables`

En fonction du temps disponible...

Refaire une manipulation similaire sur les postes terminaux avec `iptables`.

5 Utilisation de `ssh` : connexions sécurisées

On va utiliser OpenSSH en privilégiant le protocole version 2 plus sécurisé. On supprimera le protocole version 1 du fichier de configuration globale car c'est un protocole qui a intrinsèquement de nombreuses failles connues.

La commande `ssh` permet de se connecter à distance ou de lancer des commandes à distance.

La commande `scp` permet de copier des fichiers à distance. La commande `sftp` fournit une interface FTP pour les nostalgiques.

³Dangereux pour l'ordinateur, voire pour le rendement de l'utilisateur... ☺

⁴Sachant que de toute manière un utilisateur pourra toujours avoir un modem dans son bureau ou tirer un tunnel sur n'importe quel protocole autorisé et ouvrir un canal caché... Donc rester raisonnable dans les restrictions et plutôt convaincre les utilisateurs.

5.1 Connexion à distance

Se connecter via `ssh` sur une machine distante d'un collègue.

Remarquez qu'à la première utilisation `ssh` dit qu'il ne connaît pas la clé publique du `sshd` de la machine distante et l'affiche.

C'est typiquement le problème de l'authentification par mécanisme public et l'œuf ou la poule : comment être sûr que le `sshd` qu'on veut atteindre est bien celui auquel on pense et qu'on n'est pas redirigé par un moyen quelconque vers un `sshd` pirate qui espionnera tout ce que vous faites ?

C'est justement cette clé privée distante qui permet de vérifier ça. Si vous vous êtes déjà connecté sur votre machine via `ssh`, `~/ .ssh/known_hosts2` contient déjà cette clé et `ssh` reconnaissant une machine amie ne vous posera pas la question précédente. Si votre connexion est détournée, `ssh` refusera la connexion car la clé renvoyée par le `sshd` distant ne sera probablement pas celle de `~/ .ssh/known_hosts2`.

Mais revenons au problème de la question initiale de « la première fois » : comment être sûr que c'est la bonne clé qui est envoyée et donc que c'est bien la même machine ? Plusieurs solutions :

- vous connaissez la clé par cœur et vous voyez bien que c'est la même ☺ ;
- vous l'avez sur une disquette, un CDROM, une carte ou un ruban perforée⁵ ;
- vous avez coupé en 3 votre clé : un bout est gravé sur votre chaussure gauche, un autre est brodé à l'intérieur de votre slip et enfin le dernier morceau est tatoué en haut de votre nuque ;
- vous interprétez la signature de la clé en binaire et faites le *piercing* correspondant (entre 0 et 128 trous) ;
- vous n'avez pas de solution et vous acceptez de vous connecter après avoir fait une prière.

Essayer d'espionner les paquets qui passent avec `tcpdump`, `ethereal` ou `snoop` (sous Solaris). Comparer à une connexion `telnet` ou `rlogin`.

À partir d'ici oublier que `telnet`, `rlogin` ou `ftp` aient pu exister ☺ et ne les utilisez plus que sous la menace.

5.2 Création d'un couple de clés

Se créer un joli couple de clé publique-clé secrète protocole version 2 avec

```
ssh-keygen -t dsa
```

que l'on chiffrera avec la belle phrase secrète demandée.

Afin de permettre les connexions `ssh` depuis une machine \mathcal{A} vers une machine \mathcal{B} on mettra le contenu du `~/ .ssh/id_dsa.pub` sur \mathcal{A} dans le contenu `~/ .ssh/authorized_keys` de \mathcal{B} . Ainsi les connexions des utilisateurs possédant le `~/ .ssh/id_dsa` correspondant pourront se connecter à \mathcal{B} .

Pour que `sshd` soit content dans sa grande paranoïa altruïste et vous laisse vous connecter, faire un

```
chmod go-rw ~/ .ssh/authorized_keys
```

Vérifiez que cela fonctionne depuis \mathcal{A} avec un

```
ssh  $\mathcal{B}$ 
```

⁵Évidemment, l'intérêt en France est que la majorité des gens capables de décoder ça sont à la retraite, mais méfions-nous des espions soviétiques... ☺

qui va vous demander la phrase secrète de la clé DSA et non plus le mot de passe UNIX.

Si on fait une utilisation intense du réseau (connexion à plein de machines en continu, mises à jour avec des serveurs CVS via `ssh`, utilisation d'ordinateurs parallèles avec du MPI sur `ssh`, etc.), on est amené à taper sans arrêt sa (ses) phrase(s) secrète(s), ce qui va rebuter n'importe quel(le) informaticien(ne)⁶. C'est pour cela qu'a été créé le serveur `ssh-agent` gardien suprême des clés qui, une fois lancé et instruit des clés, va répondre à votre place.

Lancer dans une fenêtre un⁷

```
eval `ssh-agent`
```

qui a pour effet de lancer `ssh-agent` d'une part et d'initialiser des variables du `shell`⁸ qui donneront aux futurs `ssh` le moyen de se connecter à l'agent d'authentification.

On utilise `ssh-add` pour rajouter des clés secrètes dans l'agent. Sans paramètre supplémentaire c'est la clé DSA qui sera rajoutée par défaut avec :

```
ssh-add
```

Essayer de vous connecter sans mot de passe depuis \mathcal{A} vers \mathcal{B} avec un

```
ssh  $\mathcal{B}$ 
```

5.3 Le protocole d'affichage X11

5.3.1 Comprendre X11

X11 est un protocole permettant de faire des affichages graphiques à distance ce qui est extrêmement pratique et est ce qui est typiquement utilisé dans le monde UNIX au niveau du fenêtrage.

Par défaut les applications (clientes X11) essayent de se connecter à l'écran de la machine locale (le serveur X11) mais si on définit une variable d'environnement `DISPLAY` ou qu'on joue avec une option d'exécution on peut rediriger l'endroit d'affichage vers l'écran dont on indique le nom, ce qui est extrêmement puissant :

- télétravail ;
- on fait tourner un programme sur *la* machine qui le permet et on affiche sur une autre ;
- architecture système simplifiée : gros serveur d'applications et affichage sur des clients légers (terminaux X11,...);
- ...

Le format d'un nom d'écran X11 est du style

```
machine:écran.sous-écran
```

avec *écran* et *sous-écran* des entiers naturels commençant à 0 et le protocole sur IP/TCP utilise le port $6000 + \text{écran}$.

Dans une fenêtre afficher la valeur par défaut du nom d'écran utilisé avec par exemple :

```
echo $DISPLAY
```

⁶Ne devient-on pas informaticien(ne) par flemme, pour faire travailler les ordinateurs à sa place ?

⁷Regardez bien le modèle d'apostrophe dans ce qui suit et comprendre le *pourquoi* du *comment*.

⁸Pour bien faire il faut arranger son environnement de travail pour que `ssh-agent` soit lancé lors de votre connexion à votre ordinateur et que toutes les fenêtres en héritent.

5.4 X11 vous suit partout via ssh

Se connecter sur une machine distante avec `ssh` et lancer une application X11 (par exemple `xclock`) sans rien toucher à la variable d'environnement `DISPLAY`. Miracle !

Regarder ce qui passe entre les deux machines. Mais par où passe donc le protocole X11 ?

Afficher la valeur de `$DISPLAY` dans la fenêtre `ssh` et dans la fenêtre de votre machine locale. Essayer de comprendre le principe utilisé.

5.5 Mieux comprendre X11

Partie pour les plus curieux...

Sur la machine distante, dans la fenêtre `ssh` ou autre (`rlogin`, `telnet`,...), supposons qu'on n'utilise pas le mode mandataire X11 de `ssh` et faisons un

```
DISPLAY=votre-machine:0
```

et relancer l'application X11 précédente. Que se passe-t-il ? Faire alors sur la machine locale un

```
xhost +machine-distante
```

qui veut dire « autorise **toutes**⁹ les connexions distantes X11 sur mon écran depuis *machine-distante* ». Relancer l'application X11 précédente en regardant ce qui passe sur le réseau.

Exemple de danger : demander à un collègue de faire alors un

```
ssh machine-distante xwd -root -display votre-machine:0 | xwud
```

Expliquer comment cela marche et pourquoi¹⁰. Regarder éventuellement ce qui passe sur le réseau.

Bon, on supprime ce trou de sécurité sur sa machine en revenant en arrière avec

```
xhost -machine-distante
```

Notons que comme trou de sécurité on aurait pu faire pire si on avait carrément fait dans la mondialisation globale :

```
xhost +
```

qui signifie d'« autoriser les accès X11 depuis n'importe où » ! Évidemment à n'utiliser que sous la torture...

Une sécurisation intermédiaire si on n'a pas `ssh` sous le coude serait d'utiliser l'authentification par secret partagé au moyen de `xauth`. On peut obtenir la liste des secrets avec

```
xauth list
```

et en rajouter par exemple sur la machine distance avec

```
xauth add ma-machine:0 MIT-MAGIC-COOKIE-1 mon-secret-hexadécimal
```

⁹C'est à dire *pas que* les vôtres !!! ☺

¹⁰Exemple de la souplesse et la beauté d'UNIX au passage.

Si ce sont des machines qui partagent votre répertoire principal et en particulier le fichier `~/.Xauthority` qui a le bon goût de stocker les secrets de `xauth` vous n'aurez pas à vous en préoccuper. Sinon, il y a aussi des scripts tels que `xrsh` qui ouvrent une connexion à distance par `rsh`, positionnent le `DISPLAY` et font le `xauth` qui va bien. C'est automatique mais beaucoup moins sécurisé que `ssh`.

Philosopher sur le sens de la vie... Réfléchir à la facilité qu'apporte `ssh` et `X11` : non seulement c'est simple mais en plus c'est sécurisé !

Évidemment, si la machine distante est compromise et que vous utilisez l'encapsulation de `X11` dans `ssh` un pirate peut accéder à vos fenêtres `X11` sur votre machine locale, ce qui est fâcheux. Dans ce cas, il est clair que ce n'est pas une bonne idée d'utiliser cette encapsulation. C'est donc un compromis à trouver par machine.

5.6 Créer un Intranet sécurisé

On se propose d'utiliser ici `ssh` pour faire des connexions `WWW` sécurisées et d'autre part d'encapsuler le protocole peu fiable `telnet` en entrée. Pour ce faire on va utiliser le mécanisme de télé-détournement de port `TCP` de `ssh`, à savoir user des options `-R` et `-L` lors d'une connexion à distance.

On va supposer ici que vous faites du télétravail et que le `PC3` est chez vous. On veut pouvoir accéder directement aux serveurs `WWW` internes de l'entreprise accessibles depuis la machine interne `PC1`. On a aussi envie, si vous avez laissé allumé chez vous votre `PC3`, de se connecter depuis son bureau avec `PC2` sur votre `PC3` hyper-protégé dont même le port `ssh` est filtré.

Essayer par exemple de faire qu'une connexion `ssh` (en mode verbeux) du `PC3` vers le `PC1` vous permette de faire suivre une connexion `TCP` port 8080 sur le `PC3` vers `proxy.ensmp.fr` port 8080 (mandataire `WWW`) et sur le port 12345 de `PC1` vers le port 22 (`ssh` !) de `PC3`. **Éventuellement faire un schéma pour comprendre les encapsulations à réaliser.**

Configurer son navigateur `WWW` sur `PC3` pour utiliser le mandataire (*proxy* en anglais) `WWW` distant via `ssh`.

Se connecter avec `ssh` depuis le `PC2` vers le `PC3` en passant par le `PC1`. Le truc qui aide : regarder l'option `-p` de `ssh`...

Sur ce concept on peut créer des tunnels avec n'importe quel protocole en exécutant par exemple des `PPP` à distance (les passionnés iront voir par exemple le `TP` sur `PPP`), voire en plus du `ssh` dans d'autres `ssh`, etc. comme vu précédemment.

5.7 Et `ssh` sous Windows ?

Eh bien c'est possible ! Si on avait le temps on pourrait voir par exemple sur http://www.hsc.fr/ressources/breves/remote_windows_ssh.html.

6 SSL et navigateur `WWW`

Se connecter à un site sécurisé par `SSL` avec votre navigateur `WWW` et analyser le certificat utilisé par le site.

Regarder la liste des certificats de confiance installés avec le navigateur.

7 Journaux d'information

Regarder les messages de log avec les commandes adéquates.

8 Chiffrement de documents, de courriel et signature avec GPG

Nous allons nous familiariser ici avec le chiffrement de documents et les concepts associés avec le logiciel GPG (<http://www.gnupg.org>).

8.1 Chiffrer pour soi un document

Le mode le plus simple de fonctionnement de `gpg` est de pouvoir chiffrer un document pour soi avec un algorithme symétrique utilisant une phrase secrète et de pouvoir relire le document avec cette même phrase.

Chiffrer un document avec

```
gpg --symmetric doc
```

qui crée par défaut un `doc.gpg` et déchiffrez-le ensuite avec

```
gpg --decrypt doc.gpg
```

8.2 Démarrage

Passons aux choses plus compliquées...

Sur <http://www.gnupg.org/gph/fr/manual.html> on peut trouver une introduction plus étendue qu'ici.

8.2.1 Créer ses clés

Créer son couple de clé publique-clé privée avec

```
gpg --gen-key
```

en choisissant par exemple les options par défaut. La phrase secrète a été utilisée pour chiffrer votre clé privée. Ainsi même si quelqu'un lit vos fichiers il ne la récupèrera pas.

Cela a généré des chose dans `~/ .gnupg`.

8.2.2 Créer un certificat de révocation

Tant qu'à faire on va créer un certificat de révocation qui servira en cas de vol de votre clé secrète à annuler les copies de votre clé publique qui seront disséminées sur Internet.

```
gpg --gen-revoke votre-nom
```

On pourrait imaginer imprimer ce certificat et le cacher (pourquoi ?). Il servirait pour annuler votre clé au cas où vous n'auriez plus accès à votre ordinateur.

8.3 Gérer son trousseau de clés publiques

Vous pouvez afficher son contenu avec

```
gpg --list-keys
```

Envoyez votre clé publique à un(e) collègue pour qu'il puisse ensuite vous envoyer un courriel chiffré. Pour se faire il faut exporter une de vos clés publiques, par exemple celle avec votre *nom*, avec

```
gpg --armor --output ma-clé.gpg --export nom
```

Vous pouvez lui envoyer par un moyen sécurisé¹¹ ce fichier *ma-clé.gpg*.

Récupérez la clé publique d'un(e) collègue et intégrez-la à votre porte-clé avec

```
gpg --import sa-clé.gpg
```

Comment être sûr que cette clé lui appartient bien ?

8.4 Envoyer un fichier chiffré ou signé

Bon, le problème est que sur le réseau de test le courriel ne fonctionne pas... Mais pas de panique, on va envoyer du courrier sous forme de fichiers comme au bon vieux temps, mais avec `scp` par exemple.

Envoyer à un(e) collègue qui vous aura envoyé sa clé publique au préalable un courriel chiffré avec

```
gpg --encrypt --recipient votre-collègue document
```

Recevez un courriel (fichier) chiffré que vous lirez avec

```
gpg --decrypt document.gpg
```

Créez une signature à un document avec :

```
gpg --clearsign document
```

qui va créer un *document.asc* contenant le document suivi de la signature. Il ne reste plus qu'à envoyer ce fichier au destinataire.

Vérifier la signature d'un courriel non chiffré :

```
gpg --verify document.asc
```

Il y a aussi d'autres modes de signature (signature dans fichier séparé, etc.).

8.5 Conclusion

Vous connaissez maintenant les concepts d'usage d'un système de chiffrement et de signature de fichiers !

Complicé ? Bah, il existe probablement sur le site de GPG un client s'adaptant à votre système de gestion de courrier favori.

Il existe une interface intégrant PGP dans Emacs pour simplifier la tâche et aussi des systèmes PGP sous Windows avec fenêtres et tout.

¹¹Sinon un pirate pourrait au passage la remplacer par sa propre clé publique...

9 Mots de passe jetables avec OPIE

Pas testé récemment mais marchait sous Solaris... Pour les fanatiques, mais comme expliqué en cours mieux vaut se cantonner à du ssh quand on peut...

Récupérer le logiciel opie sur `ftp://ftp.inner.net/pub/opie` ou utiliser la version se trouvant dans `/usr/local/src/Security`.

Créer un répertoire `~/TP/Securite` pour travailler.

Y décompresser opie avec `gtar zxvf`.

Faire un

```
./configure --prefix=/usr --enable-insecure-override  
make
```

L'option `-enable-insecure-override` indique à opie de permettre un calcul même s'il pense que c'est non sécurisé : mot de passe passant à travers le réseau. À utiliser avec extrême prudence dans des cas où on pense qu'opie est paranoïaque.

L'option `--prefix=/usr` permet de faire l'installation dans `/usr` au lieu du normal `/usr/local` partagé car dans le mastère nous voulons que chacun installe le logiciel.

Sous root faire un `make install`.

Entrer un mot de passe via `opiepasswd`.

Penser à le faire aussi pour root avant de se déconnecter...

Tester via `login`, `ftp`, `su` et `telnet`¹².

Sortir une page de secrets pour pouvoir se connecter sans « calcullette ».

Pour désinstaller OPIE il suffit normalement de faire

```
make uninstall
```

¹²Ce n'est pas la peine d'utiliser `rlogin` : `rlogin` utilisant `hosts.equiv` pour ne pas demander de mot de passe au sein du mastère ne demandera pas de mot de passe.