



Sécurité des réseaux et des systèmes

Compte rendu de travaux pratiques

Baptiste MATHUS

Table des matières

I Introduction.....	3
II Étape 1 : Installation du système	4
II.1 Application.....	4
III Étape 2 : Configuration du système	6
III.1 Les permissions des fichiers.....	6
III.1.1 Les bits spéciaux.....	6
III.2 Rechercher les fichiers dangereux.....	6
III.2.1 Fichiers sgid ou suid.....	7
III.2.2 Fichiers en écriture pour tout le monde.....	7
III.2.3 Fichiers sans propriétaire.....	7
III.3 chattr & lsattr.....	8
III.4 Configuration des services de base.....	8
III.4.1 Apache.....	8
III.4.2 FTP.....	10
III.4.3 SSH.....	10
III.4.3.1 Se logger sans mot de passe.....	11
III.4.3.2 Quelques options de configurations.....	11
III.4.3.3 Sécuriser un protocole existant.....	11
III.5 Configuration des services de restriction d'accès.....	14
III.5.1 Options du noyau.....	14
III.5.1.1 Réponse au ping.....	14
III.5.1.2 Redirection de paquets.....	15
III.5.1.3 Marques de temps.....	15
III.5.2 Durcissement des mots de passe.....	15
III.5.3 TCP Wrapper.....	15
III.5.4 Firewall.....	17
III.5.4.1 Masquerading.....	18
IV Étape 3 : Auditer la sécurité d'un système.....	19
IV.1 Lister les services avec Nmap.....	19
IV.2 Auditer la sécurité des services d'une machine avec Nessus.....	21
IV.2.1 Test.....	21
V Conclusion personnelle.....	24

I Introduction

Ce document est un compte-rendu de travaux pratiques réalisé en 3^e année d'école d'ingénieurs chez Ingénieurs 2000 en filière Informatique Réseaux pendant le cours de *sécurité des réseaux* de Sébastien Lelong et Yves Lavanant.

Il présente les différents principes et étapes de la sécurisation d'un système. Au maximum, on s'efforcera de ne pas seulement répondre aux questions et ainsi contraindre le lecteur à jouer à Jeopardy. On tentera d'apporter des explications claires et complètes, présentant la problématique avant d'en présenter la solution.

Dans un premier temps sera présentée l'installation du système. On consacra ensuite la plus grosse partie du compte-rendu aux étapes de la configuration de la machine. On terminera par la présentation de l'audit de système.

II Étape 1 : Installation du système

Lors de l'installation d'un système serveur, il convient de n'installer que le strict minimum sur la machine. Seuls les services utilisés doivent être activés et installés. La machine doit donc être épurée de tous les outils, packages ou services qui ne seront pas utiles et utilisés.

L'un des exemples les plus évidents est le serveur X. Il est d'une totale incohérence de laisser installé une telle application sur une machine serveur. C'est une faille potentielle dont se passerait très bien un serveur de production. Ce type de machine destinée à être simplement allumée, sans écran parfois, n'a aucun besoin de ce type d'application.

Si on désire contrôler la machine, l'installation d'un serveur ssh correctement sécurisé¹ répondra beaucoup mieux à ce besoin.

II.1 Application

Les machines des salles de travaux pratiques sont des debian pré-installées. Pour lister les packages installés, la commande à utiliser est la suivante :

```
# dpkg -l
```

Comme indiqué ci-dessus, l'un des premiers packages à supprimer sur un serveur en production est le serveur X. On peut remarquer que la machine de test contient déjà plusieurs packages totalement inutiles :

```
# dpkg -l |grep X11
ii  fluxbox          0.9.9-1      Highly configurable and low resource X11 Win
ii  gnuplot-x11      4.0.0-2      X11-terminal driver for gnuplot
ii  imlib1           1.9.14-16    imaging library for X and X11 (using libpng2
ii  kdevelop         2.1.5.1-7    An IDE for Unix/X11
rc  libsd11.2debias 1.2.7-10     Simple DirectMedia Layer (with X11 and OSS o
ii  tk8.3            8.3.5-4      Tk toolkit for Tcl and X11, v8.3 - run-time
ii  tk8.4            8.4.7-1      Tk toolkit for Tcl and X11, v8.4 - run-time
ii  unifont          1.0-1        X11 dual-width GNU unicode font
rc  xine              0.4.3-2      MPEG, VCD, DVD audio/video player for X11
ii  xpvt-common      0.0.9.final.00 Xprint - the X11 print system (configuration
ii  xpvt-xprintorg  0.0.9.final.00 Xprint - the X11 print system from xprint.or
```

La distribution intègre un système de gestion de paquets très évolué apt-get. D'aucuns diraient même le plus évolué de toutes les distributions, mais « ne nourrissons pas le troll :-) ».

¹ N'autorisant que les connexions depuis une machine précise par exemple.

Exemple de désinstallation du paquet kdevelop avec debian :

```
# apt-get remove kdevelop --purge
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Les paquets suivants seront ENLEVÉS :
  kdevelop* kdevelop-data*
0 mis à jour, 0 nouvellement installés, 2 à enlever et 18 non mis à jour.
Il est nécessaire de prendre 0o dans les archives.
Après dépaquetage, 5222ko d'espace disque seront libérés.
Souhaitez-vous continuer ? [O/n]
(Lecture de la base de données... 168194 fichiers et répertoires déjà
installés.)
Suppression de kdevelop ...
Purge des fichiers de configuration de kdevelop ...
Suppression de kdevelop-data ...
```

En résumé, tout paquet qui n'est pas d'une utilité avérée pour le rôle du serveur, dont on n'a pas vérifié qu'il était à jour en ce qui concerne les failles connues, devra être systématiquement supprimé du système.

III Étape 2 : Configuration du système

III.1 Les permissions des fichiers

Sur un système d'exploitation de type Unix, les permissions des fichiers s'expriment grâce à 12 bits regroupés par groupes de 3. Les 4 groupes sont organisés comme suit :

Spécial			Droits de l'utilisateur			Droits du groupe			Droits des autres		
suid	sgid	sticky	read	write	execute	read	write	execute	read	write	execute
4	2	1	4	2	1	4	2	1	4	2	1

III.1.1 Les bits spéciaux

- Suid : s'il est positionné, alors l'exécutable se lancera avec les droits de l'utilisateur à qui il appartient.
- Sgid : de même, s'il est positionné, alors l'exécutable se lancera avec les droits du groupe auquel il appartient.
- Sticky : historiquement ce bit était utilisé pour demander à l'OS de conserver le plus possible un exécutable en RAM. À présent, on utilise plutôt le sticky bit sur un répertoire. Lorsqu'il est positionné et que tout le monde a le droit en écriture, tout le monde peut effectivement écrire dans le répertoire, mais seul celui qui a écrit un fichier peut l'effacer. Ce bit est utilisé sous Linux pour un répertoire très connu : /tmp.

```
# ls -l -d /tmp
drwxrwxrwt 13 root root 11264 2004-12-24 21:08 /tmp
```

III.2 Rechercher les fichiers dangereux

Certains fichiers peuvent se révéler dangereux pour la sécurité du système si leurs permissions sont mauvaises. Par exemple, si un exécutable dont on n'est pas sûr de l'écriture possède un suid bit et que celui-ci appartient à root, il devient possible à un attaquant de devenir root sur le système. Un problème identique se pose si le sgid bit est positionné et que le groupe de l'exécutable est celui d'un groupe qui possède de grands privilèges sur le système.

D'une manière générale, on évitera de positionner ce type d'attribut si ce n'est pas absolument indispensable (comme pour le programme passwd qui a besoin d'avoir le suid bit tout en appartenant à root pour pouvoir modifier /etc/passwd par exemple).

III.2.1 Fichiers sgid ou suid

Pour rechercher les fichiers dont le suid ou le sgid bit est positionné, la commande find va nous être très utile :

```
# find / -perm +6000 -ls
```

L'option perm permet évidemment de trouver les fichiers dont les permissions sont spécifiées. La partie intéressante réside dans l'utilisation du paramètre + devant le 6000. Dans la page manuel de find, on peut lire :

```
-perm +mode
```

```
Fichier ayant certaines des autorisations indiquées dans le mode.
```

On trouve ainsi simplement tous les fichiers qui ont soit le bit sgid (2) soit le bit suid (4) positionné (soit les deux).

III.2.2 Fichiers en écriture pour tout le monde

Un fichier ayant ce type de droits est modifiable par tous. C'est rarement quelque chose que l'on souhaite. Dans la plupart des cas, il s'agira d'un utilisateur, ou pire d'un administrateur, qui aura fait un mauvais usage de la commande chmod.

Il faut surveiller et limiter au maximum ce type de fichier. En cas attaque, si l'objectif est par exemple de faire tomber un service, il suffira d'écrire indéfiniment dans ce type de fichier pour remplir la partition dans laquelle il se trouve.

Si le fichier en question se trouve sur une partition utilisée par le service (où ce dernier écrit ses logs par exemple), au mieux les logs ne pourront plus être réalisés, au pire le service s'arrêtera tout simplement...

Pour trouver ces fichiers :

```
find / -perm +0002 -and -type f -ls
```

III.2.3 Fichiers sans propriétaire

```
find / -nouser -or -nogroup
```

Ces fichiers peuvent poser des problèmes parce qu'ils pourraient être réaffectés à un nouvel utilisateur si celui-ci récupère l'uid correspondant lors de sa création. Il est donc sage de conserver une trace claire de l'appartenance des fichiers lorsqu'un employé quitte une entreprise par exemple.

III.3 chattr & lsattr

Ces deux commandes permettent de contrôler des attributs de fichiers gérés directement au niveau du système de fichier. Ceci permet notamment de se protéger des scripts-kiddies qui seront rapidement repoussés par ce problème. Un des intérêts est aussi de se protéger de soi-même : même un `rm -f` en tant que root sur un fichier protégé via cette commande ne sera pas autorisé.

Exemple avec l'attribut append (ne permet que l'ajout de données en fin de fichier) :

```
# whoami
root
# touch pouet
# chattr +a pouet
# echo "bonjour" > pouet
bash: pouet: Opération non permise
# echo "bonjour" >> pouet
# cat pouet
bonjour
# rm -f pouet
rm: ne peut enlever `pouet': Opération non permise
# lsattr pouet
-----a----- pouet
```

III.4 Configuration des services de base

III.4.1 Apache

Apache est une application fréquemment utilisée. C'est le logiciel de serveur web le plus utilisé dans le monde et réputé le plus sûr. Il n'en reste pas moins qu'il est important de correctement le configurer pour la production pour bénéficier de cette « sûreté » au risque d'obtenir l'effet inverse.

Il est possible de lancer Apache avec `(x)inetd` dont on verra l'utilisation plus loin, mais ceci rarement utilisé. En effet, il ne paraît pas justifié d'arrêter et de relancer chaque fois un serveur Apache lorsqu'une requête parvient au serveur. Celui-ci intègre de plus une fonction avancée de gestion des requêtes, configurable à souhait.

Deux paramètres sont particulièrement importants :

– ServerSignature :

Ce paramètre peut valoir *On*, *Off* ou *Email*. La documentation écrite dans le fichier de configuration est très claire.

Le serveur l'utilise pour les pages qu'il va générer pour rajouter ou non une ligne en bas de page indiquant son numéro de version ainsi que les modules chargés. Il faut le positionner à *Off* pour renvoyer le moins d'informations possibles sur le serveur.

Avec *on*

```
Apache/1.3.33 Server at localhost Port 80
```

Avec *Email*, l'adresse de l'administrateur est rajoutée en lien hypertexte sur localhost.

Avec *off*, cette ligne n'apparaît plus.

– ServerTokens

Ce paramètre peut prendre 5 valeurs. Dans les informations http renvoyées, celles-ci vont conditionner le niveau d'information renvoyé par le serveur.

Pour un serveur en production, on cherche à renvoyer le moins possible d'information. On va donc positionner ce paramètre à celui le moins informatif : *prod*.

Sans ce paramètre, on reçoit la chaîne suivante :

```
Server: Apache/1.3.33 (Debian GNU/Linux) PHP/4.3.10-2 mod_ssl/2.8.22  
OpenSSL/0.9.7d
```

Avec *ServerTokens Prod* :

```
Server: Apache
```

Afin de tendre vers une sécurité maximale, on peut envisager des solutions supplémentaires à celles fournies de base. Celles-ci ne seront pas détaillées ici, on se contentera d'en donner une brève description.

chroot : On peut faire tourner Apache, comme n'importe quel programme dans une « prison logicielle ». Ainsi, même si l'attaquant parvient à s'introduire sur la machine par le biais du service, il ne pourra pas remonter à la véritable arborescence du système et restera enfermé dans le chroot².

UML (User Mode Linux) : On pousse encore plus loin le concept de prison logicielle. Ici, c'est le système d'exploitation complet qui tourne comme un processus normal au sein du véritable système d'exploitation. Par ce biais, on peut limiter l'attaquant de telle façon à ce que même s'il arrivait à cracker l'OS complet, il ne se retrouverait finalement qu'avec un compte utilisateur sur la machine et le véritable système d'exploitation qui fait fonctionner celui qu'il a piraté.

Xen : Un produit analogue à User Mode Linux, mais qui pousse le concept encore plus loin. Ce logiciel a collaboré avec Microsoft pour réaliser une version qui soit utilisable comme un processus au sein de cet OS. Ainsi, au sein d'un Xen, il est possible de faire à la fois fonctionner un Linux et un Windows en même temps.

III.4.2 FTP

ProFTPD est un serveur récent plus récent que le plus connu Wu-Ftpd. On l'utilisera parce qu'il semble plus performant et présenter moins de failles de sécurité.

Sur ce type de serveur, il est intéressant de fournir un répertoire *Incoming* afin de permettre le dépôt de fichier. Toutefois, il est extrêmement important d'en interdire la récupération immédiate (au moins avant modération d'un administrateur). Si on ne procède pas ainsi, on s'expose à devenir un lieu de dépôts et d'échanges de fichiers piratés.

Le plus souvent, on autorise donc uniquement le PUT dans ce répertoire et on ne permet pas le listing (ls) du répertoire.

III.4.3 SSH

Secure Shell est une alternative récente à d'anciens systèmes de connexion comme telnet ou rsh. La grande différence réside dans sa capacité à chiffrer les échanges en utilisant un système de clés asymétriques comme RSA1, RSA2, DSA, AES...

² chroot est une commande qui permet de changer la racine du système de fichier (/). C'est grâce à elle qu'on peut empêcher un pirate de toucher à d'autres endroits que celui par lequel il s'est introduit.

III.4.3.1 Se logger sans mot de passe

Grâce à ssh, il est possible de se logger sur la machine distante sans fournir de mot de passe. Pour ce faire, on utilise seulement notre clé publique. Celle-ci va s'associer à notre clé privée pour l'authentification au login et nous permettre le login sans demander quoi que ce soit. Ce principe de fonctionnement est notamment souvent utilisé pour automatiser des copies distantes sécurisées via l'utilisation de la commande scp.

III.4.3.2 Quelques options de configurations

Autoriser un nombre limité d'utilisateurs :

```
AllowUsers titi tata
```

Cette option de configuration (/etc/ssh/sshd_config) permet de n'autoriser que les utilisateurs titi et tata à se connecter à la machine via le serveur ssh.

Interdire le login distant à root

```
PermitRootLogin No
```

On doit interdire le login direct à l'utilisateur root sur la machine. Si un administrateur souhaite se connecter, il devra d'abord se connecter par un utilisateur « normal » et utiliser ensuite su* pour effectuer les commandes qu'ils désire.

III.4.3.3 Sécuriser un protocole existant

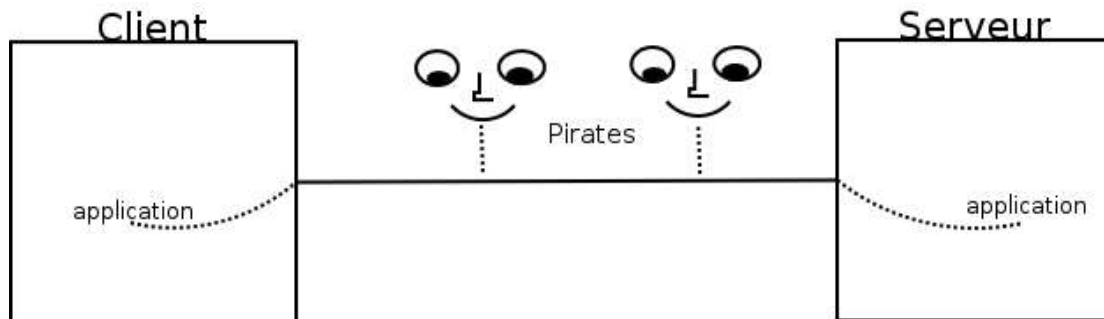
Si tout doit être sécurisé, que faire alors des protocoles ou logiciels conçus à une époque où les préoccupations de sécurité n'étaient pas aussi présentes, où le plus grand nombre n'avait pas accès simplement à Internet ? On ne peut évidemment pas se permettre de tout jeter. Il est indispensable de trouver une solution, au moins temporaire d'utilisation sécurisée.

La solution s'appelle le tunelling. On va utiliser ssh pour créer un tunnel sécurisé entre deux machines qu'on souhaite faire communiquer et simplement modifier les adresses/ports des machines à atteindre. Ceci sera transparent pour « l'ancienne » application.

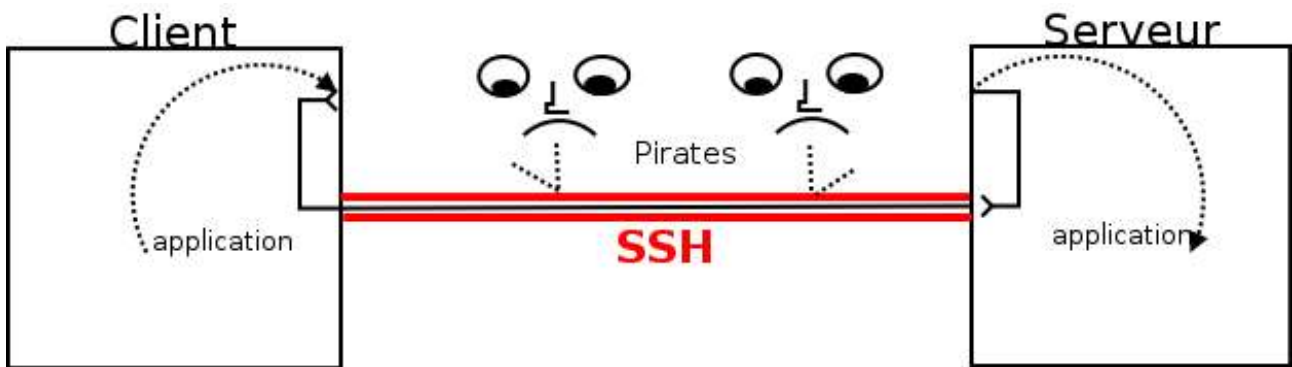
SSH permet de créer une socket d'écoute sur la machine locale sur un port donné. Tout ce qui arrivera dans le port sera retransmis via le tunnel à destination du serveur ssh distant, qui à son tour redirigera vers la machine (locale ou non) et le port demandé les paquets.

Exemple :

Sans ssh :



Avec ssh :

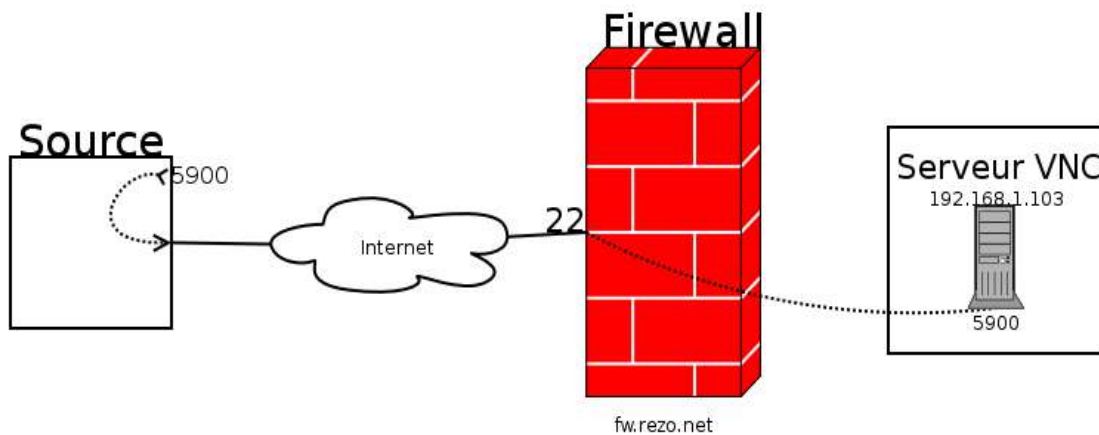


Passage de firewall

Cette solution peut aussi permettre de « passer » un firewall. Si un serveur ssh tourne sur ce dernier et qu'on y a un compte, on peut lui demander de rediriger des données vers un serveur à l'intérieur du réseau.

Exemple d'utilisation pour passer un firewall :

Comment atteindre un serveur VNC³ qui se trouve derrière un firewall nommé *fw.rezo.net* en supposant que ce firewall possède un serveur ssh sur lequel on a un compte toto ?



Il suffit d'utiliser la commande suivante :

```
ssh -2 -N -f -g -L 5900:192.168.1.103:5900 toto@fw.rezo.net
```

C'est l'option L la plus importante, c'est elle qui demande la création du tunnel en spécifiant "port local:adresse cible:port cible".

- 2 est là pour forcer l'utilisation de sshv2,
- N pour indiquer à ssh de ne pas exécuter de commande distante (typiquement pour un forwarding de port comme ici, est-il indiqué dans la page de manuel),
- f pour passer en arrière plan après le passage d'une éventuelle passphrase et
- g pour autoriser la connexion à la socket du tunnel depuis d'autres machines.

³ VNC est un logiciel de contrôle à distance. Il permet par exemple de contrôler une machine Windows ou Linux sans être physiquement devant elle. Il suffit pour cela de faire fonctionner un serveur VNC sur la machine à contrôler, de disposer du client VNC et du mot de passe pour pouvoir prendre le contrôle de la machine. Un serveur VNC écoute par défaut sur le port 5900.

III.5 Configuration des services de restriction d'accès

III.5.1 Options du noyau

Sous Linux (et BSD), il est possible de passer des options au noyau. Ceci s'effectue via le système de fichier virtuel `/proc` qui est en fait une liste de paramètres du noyau. Dans le répertoire `/proc/sys/net/ipv4` se trouvent un certain nombre de paramètres permettant de modifier le « comportement réseau » de la machine.

III.5.1.1 Réponse au ping

1. Le paramètre `icmp_echo_ignore_all` permet de demander à la machine de ne plus répondre au « ECHO REQUEST » (i.e. au ping) qui lui sont envoyés.

```
# ping -c2 localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.5 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.0 ms
```

Par défaut sous Linux, la machine est configurée pour répondre au ping.

```
--- localhost ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.0/0.2/0.5 ms
# echo "1"> /proc/sys/net/ipv4/icmp_echo_ignore_all
# ping -c2 localhost
PING localhost (127.0.0.1): 56 data bytes

--- localhost ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
```

Avec ce paramètre, plus aucune réponse (ECHO REPLY) n'est envoyée.

2. Le paramètre `icmp_echo_ignore_broadcasts` permet de demander à la machine de ne pas répondre au ping broadcast (adresse MAC `FF:FF:FF:FF:FF:FF`) qu'elle reçoit. Il est important de positionner ce paramètre à 1 afin d'éviter que la machine ne puisse par exemple servir à participer à une attaque de type DoS (Deny of Service).

Plus précisément ici, il s'agit d'empêcher les attaques DDoS (Distributed DoS). En effet, en falsifiant l'adresse MAC source d'une machine effectuant un ping broadcast, si beaucoup de machines répondent, on peut flooder une machine qu'on souhaite attaquer : cette dernière recevrait alors un nombre important d'ECHO REPLY (dont elle n'est évidemment pas le

demandeur), nombre qui peut devenir d'autant plus grand que le réseau dans laquelle cette machine se trouve est grand.

III.5.1.2 Redirection de paquets

Le paramètre *ip_forward* permet d'indiquer à la machine qu'elle peut router les paquets qu'elle reçoit. Bien entendu, ce paramètre doit être désactivé si la machine n'est pas destinée à être un routeur.

III.5.1.3 Marques de temps

Le paramètre *tcp_timestamps* indique à la machine d'activer ou non l'horodatage des paquets TCP. Il faut désactiver ces marques afin de ne pas faciliter une attaque de type « Man in the middle ». En effet, si les marques de temps sont activées, selon les OS, il devient plus ou moins facile de prévoir le prochain numéro d'ack qui va transiter. Si l'attaquant prévoit correctement ce numéro, il peut parvenir à se substituer au véritable destinataire.

III.5.2 Durcissement des mots de passe

Pour contrôler un grand nombre de paramètre d'authentification, il existe PAM : Pluggable Authentication Module. Le système de login lui passe en quelque sorte la main pour effectuer les opérations qui ont été configurées.

Ici, nous souhaitons pour l'exemple n'autoriser que les mots de passe d'au moins 8 caractères. De même, nous voulons qu'un utilisateur n'ait que 3 tentatives pour rentrer son mot de passe.

Ces fonctionnalités sont disponibles grâce au module *libpam_cracklib*. Il est installable sur la debian via le paquet éponyme.

Pour l'utiliser, il suffit de commenter toutes les lignes du fichier */etc/pam.d/common-password* et de mettre les deux suivantes :

```
password required      pam_cracklib.so retry=3 minlen=8 difok=3
password required      pam_unix.so use_authok nullok md5
```

- *retry* et *minlen* sont assez explicites pour ne pas nécessiter d'explications ;
- *difok* permet de demander à ce qu'un nouveau mot de passe soit différent d'au moins 3 caractères par rapport à l'ancien.

III.5.3 TCP Wrapper

Xinetd est le successeur d'Inetd. On appelle ce type d'application un super-serveur, elle va en effet écouter les ports de services donnés et lancer ceux-ci afin de leur passer la main lorsqu'une demande arrive.

Les objectifs sont multiples :

- gestion centralisée de la configuration des services ;
- ajouter une gestion de la sécurité à certains services n'en intégrant aucune ;
- consommation moindre de mémoire et du processeur. En effet, si un service est utilisé une fois par mois, où est l'intérêt qu'il fonctionne 24h/24 ?
- beaucoup d'autres utilités.

Dans cette partie, on se contentera de donner quelques exemples d'utilisation de xinetd. Pour plus de détails, se reporter comme toujours aux pages manuel.

Nous allons reconfigurer proFTPD pour qu'il soit lancé par xinetd. Pour ce faire, on modifie dans son fichier de configuration l'option suivante :

```
ServerType inetd
```

Note : On remarquera que le fichier de configuration de proFTPD est très similaire à celui d'Apache, ce qui est appréciable pour le connaisseur d'Apache qui peut assez rapidement appréhender la configuration de proFTPD.

Ensuite, on ajoute simplement un fichier nommé ftp qu'on copie depuis n'importe quel autre déjà présent dans le répertoire /etc/xinetd.d qu'on va modifier ensuite pour nos besoins pour le faire ressembler à ceci :

```
service ftp
{
  disable = no
  user = root
  socket_type = stream
  wait = no
  server = /usr/sbin/proftpd
  only_from 192.168.1.103
  access_times = 7:00-20:00
}
```

On voit déjà poindre la puissance et la flexibilité qu'apporte (x)inetd : access_times si on souhaite par exemple ne rendre un service disponible qu'aux heures de travail, only_from pour limiter les adresses qui ont le droit d'accéder au service. On voit clairement de plus qu'une partie non négligeable de code a été factorisée : s'il avait fallu gérer dans chaque service ce type de restriction, le code de ceux-ci aurait été inutilement complexifié et il en aurait résulté

une configuration bien moins homogène pour l'administrateur de tous ces services.

III.5.4 Firewall

Iptables permet de réaliser un filtrage très fin de ce qu'on souhaite ou non autoriser à entrer, sortir ou passer par une machine.

Comme toujours, comme ce compte-rendu ne se veut pas un How-to Netfilter, nous allons nous contenter de répondre à quelques demandes clairement identifiées pour réaliser la configuration de notre firewall.

- Accepter les flux locaux et interdire tout le reste ;
- Autoriser tous les clients à se connecter au serveur WEB de la machine en HTTP et HTTPS (rappel, nous travaillons sur un serveur WEB).
- Une seule machine sera autorisée à accéder au FTP (192.168.1.103)

```
#!/bin/sh

#Un script flush pour remettre tout à ACCEPT
#et vider toutes les règles
./flush

#La machine n'est pas un routeur
echo "0" > /proc/sys/net/ipv4/ip_forward

#On droppe tout par défaut en entrée
iptables -P INPUT DROP

#On accepte les paquets pour localhost
iptables -A INPUT -i lo -j ACCEPT

#web (http et https)
iptables -A INPUT -i eth0 -p TCP --dport http -j ACCEPT
iptables -A INPUT -i eth0 -p TCP --dport https -j ACCEPT

#On n'autorise qu'une seule machine à accéder au serveur ftp
#iptables -A INPUT -i eth0 -p TCP --dport --source 172.17.2.5 ftp -j ACCEPT
#iptables -A INPUT -i eth0 -p TCP --dport --source 172.17.2.5 ftp-data -j
ACCEPT
```

```
#Règle statefull pour autoriser à entrer les paquets qui sont liés à une
#connexion établie (ESTABLISHED)
#ou liés à une connexion, mais dont le retour
#arrive sur un autre port par exemple (RELATED)
iptables -A INPUT -i eth0 -p TCP -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

III.5.4.1 Masquerading

Le masquerading est un type de NAT particulier, plus précisément de Source NAT. NAT signifie Network Address Translation, ce qui se traduirait à peu près par traduction d'adresse réseau. Le masquerading est très souvent utilisé : lorsqu'on ne dispose que d'une adresse IP sur Internet par exemple et qu'on souhaite quand même pouvoir surfer avec plusieurs machines, il est intéressant de pouvoir cacher les adresses internes derrière la machine connectée. Celle-ci est responsable du masquerading, lorsqu'une machine du réseau interne cherche à aller sur Internet, elle va remplacer l'adresse du paquet par la sienne avec son IP publique. Lorsque le paquet revient, elle connaît la corrélation entre le paquet et la machine qui le désire, elle peut donc le transmettre à la machine à l'intérieur du réseau local qui en a fait la demande.

IV Étape 3 : Auditer la sécurité d'un système

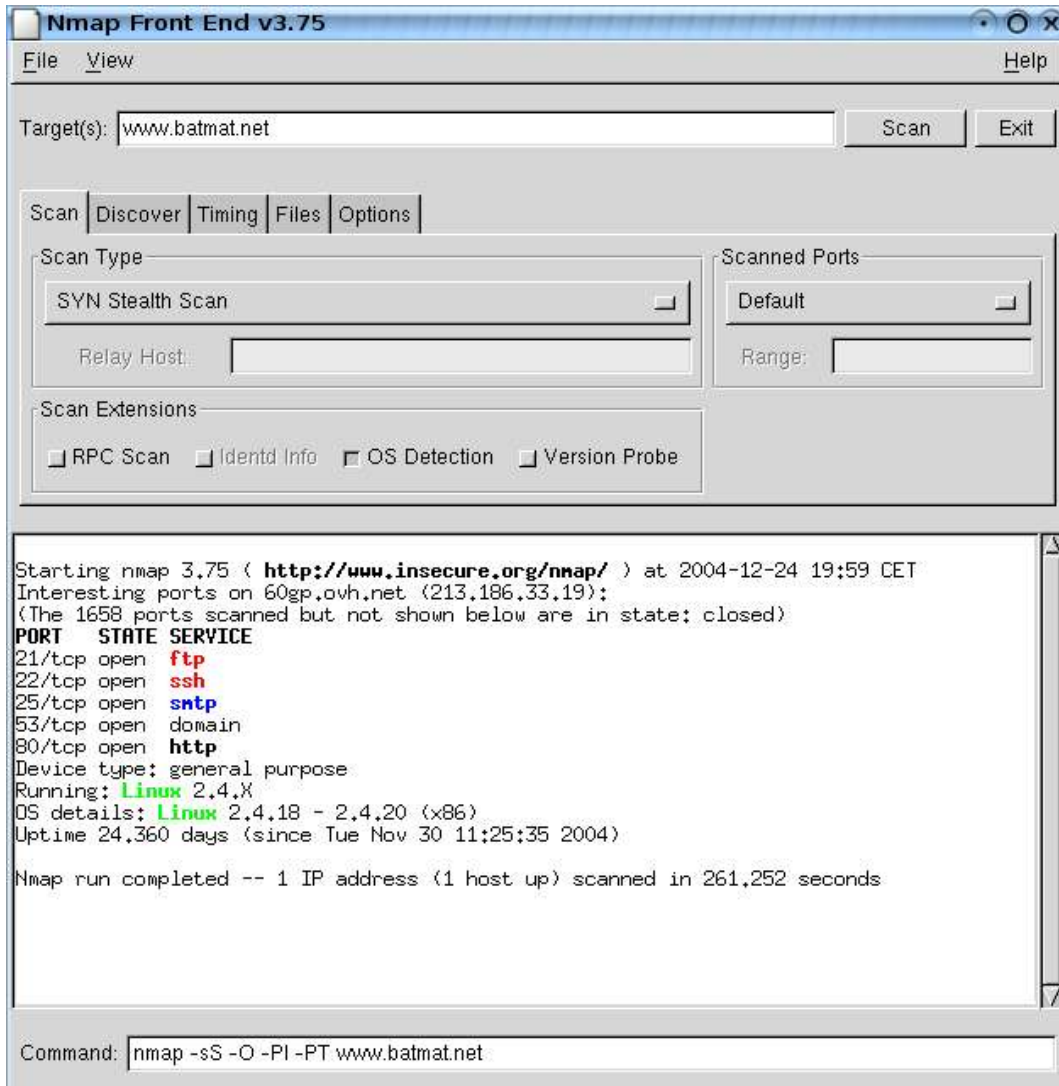
Après mise en place de différents services et procédés utilisés, il est nécessaire de vérifier leur bon fonctionnement. Il est notamment important de valider le fait que le système n'est par exemple pas sensible à de plus ou moins anciennes failles, a fortiori celles dont des exploits ont été publiés.

IV.1 Lister les services avec Nmap

Nmap est un scanner de ports. Il permet de scanner, comme son nom l'indique, les ports d'une machine pour détecter ceux qui sont ouverts, fermés ou autres. En fonction des résultats, un attaquant peut ensuite déterminer quels logiciels il va pouvoir utiliser pour tenter de pénétrer. C'est le premier type d'outil utilisé pour préparer une attaque.

Testons le sur le serveur WEB <http://www.batmat.net>. Cette adresse est hébergée chez OVH. Scanner les ports d'une machine est normalement interdit, mais nous espérons qu'ils nous le pardonneront pour les besoins du tests. Nous devrions normalement ne trouver que le port 80 ouvert, très peu d'autres puisque c'est le rôle de ce serveur.

Capture d'écran avec nmapfe⁴ :



On voit ici qu'en plus de l'indispensable port 80 puisque le rôle premier de cette machine est de servir des pages web, les ports habituels⁵ des services ftp, ssh, smtp et dns sont ouverts.

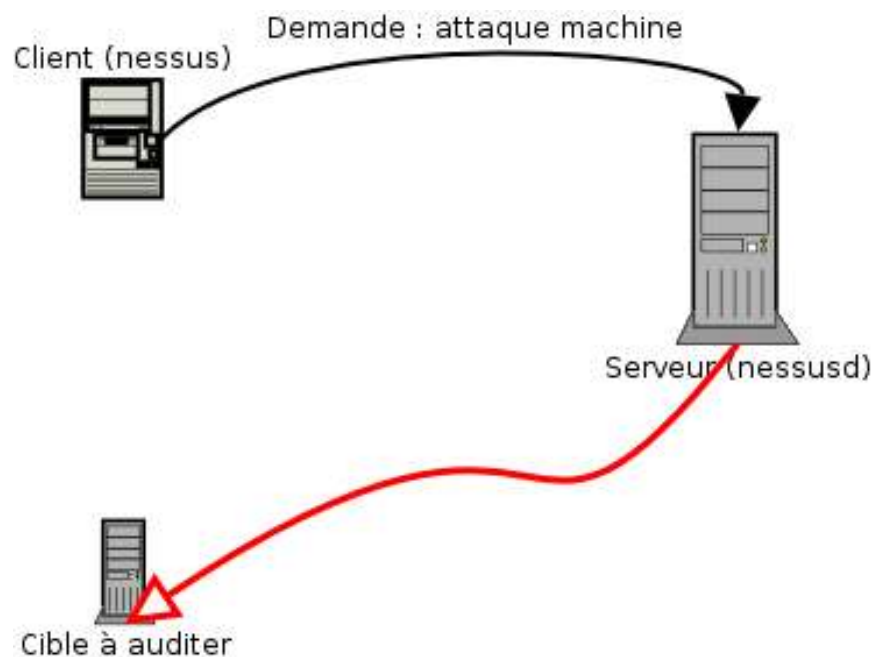
4 Nmap Front-End : un logiciel qui permet de contrôler un peu plus graphiquement les options de scan qu'on souhaite passer à nmap.

5 On utilise ici le mot « habituels » pour indiquer que ce n'est pas parce que le port 22 est ouvert que c'est forcément un serveur ssh.

IV.2 Auditer la sécurité des services d'une machine avec Nessus

Nessus est un « scanner de sécurité » comme il est défini dans sa documentation. Il permet de faire une batterie de tests de sécurité plus ou moins dangereux à une machine. Il se compose d'un client et d'un serveur, on contrôle le serveur pour attaquer n'importe quelle machine depuis le client.

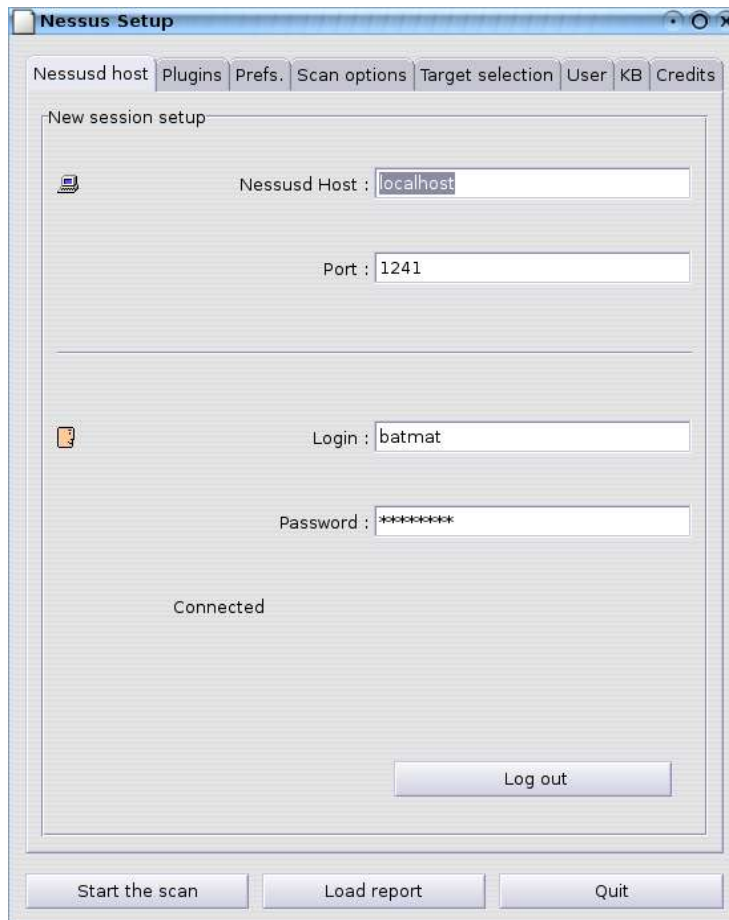
Architecture nessus :



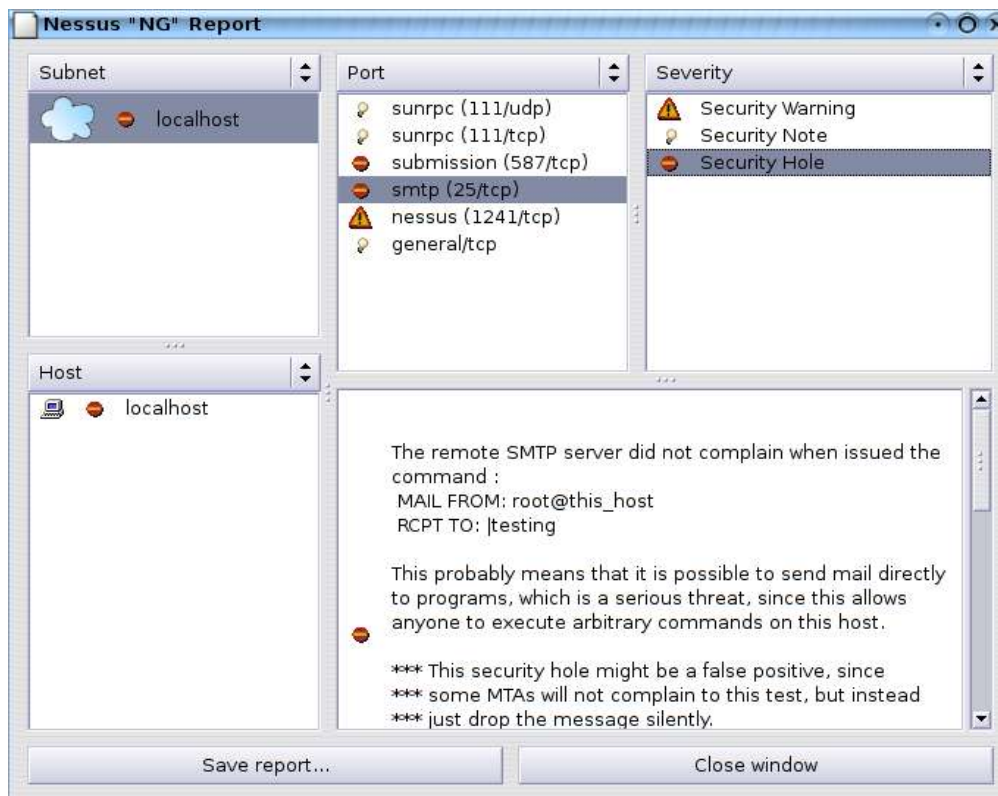
IV.2.1 Test

On effectuera ici les tests sur *localhost*. Le client, le serveur et la cible sont donc la même machine, mais le principe serait le même avec trois machines différentes.

1. On crée un compte pour le serveur `nessusd` par la commande `nessus-adduser` (attention, un user `nessusd` n'a rien à voir avec un user du système).
2. On se loggue sur le serveur via l'interface graphique du client :



3. On choisit les attaques qu'on veut réaliser dans l'onglet *Plugins*. Une option permet de ne sélectionner que les plugins non dangereux. Comme on travaille ici en local, on sauvegarde tout, on ravale sa salive et on les sélectionne tous.
4. On croise les doigts pour ne pas crasher la machine ou le disque dur (ou les deux), ce qui empêcherait vraisemblablement de rendre comme prévu le présent document avant le 24 décembre et on lance le scan (bouton *Start the scan*) après avoir indiqué la cible (localhost) dans l'onglet *Target selection*.
5. Le test s'est bien déroulé et nessus affiche un résumé des résultats :



V Conclusion personnelle

Ce TP m'a permis d'apprendre beaucoup au sujet de la sécurité d'une machine. J'ai pu approfondir ou clarifier un grand nombre de points comme iptables ou xinetd. Ce compte-rendu constituera un très bon support de base à l'avenir si j'ai besoin de configurer une machine dans un environnement sensible.