Procédure d'installation GNU/Linux Debian

SERVEUR FREERADIUS

XXX/XX/DT – 23/01/2009

Procédure d'installation

Légende : Le texte explicatif est **sous** l'image. Texte normal : explication *Texte en italique* : options à choisir, sélectionner ou valeur à rentrer dans un champ **… Texte en gras avec 3 points avant et après …** : traitement de la machine qui dure quelques secondes ou quelques minutes ; patienter jusqu'à la fin du traitement.

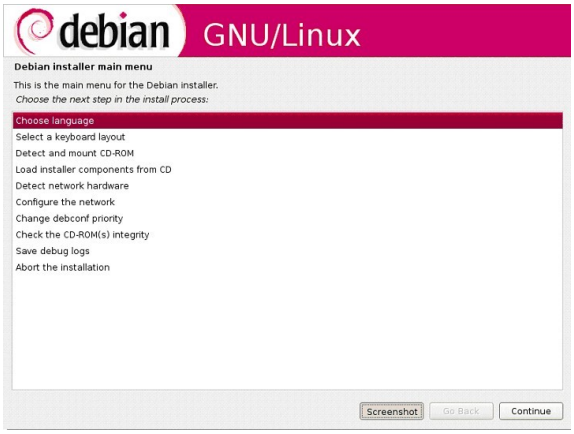
Pré-requis :

- Avoir lu la documentation en entier au-moins une fois.
- Le disque dur du serveur doit être vierge, sans partition.
- Connaître l'adresse IP, le masque sous-réseau et la passerelle du serveur
- Connaître les adresses IP des serveurs DNS

Durée :
L'installation de GNU/Linux Debian Etch dure entre 15 (habitué) et 30 (première fois) minutes.
La configuration pour aboutir à l'outil final dure entre 45 min (habitué) et 3h (première fois).

C'est parti :
Démarrer le serveur sur le cédérom.

Dans l'invite pseudo-graphique, taper *expertgui*. Attention, si l'on n'a rien indiqué au bout de quelques dizaines de secondes, l'installation se lance automatiquement, mais pas dans le mode décrit dans cette documentation. Si l'affichage des menus présentés dans ce document est mauvais où si l'on a un écran noir, c'est que nous ne pouvons pas utiliser le mode graphique normal. Nous devrons alors passer par un mode pseudo-graphique. Pour cela, redémarrer le serveur sur le cédérom et à l'invite pseudo-graphique, taper *expert*. Dans ce dernier mode, on se déplace de menu en menu avec la touche de tabulation et on sélectionne/désélectionne avec la touche d'espace.



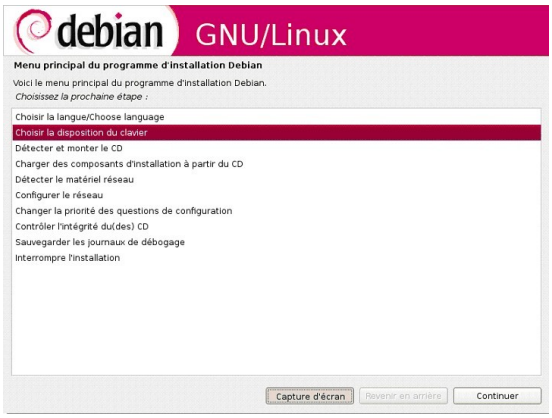
Appuyer sur *Continue*.

Sélectionner *French* et appuyer sur *Continue*.

Appuyer sur *Continuer*.

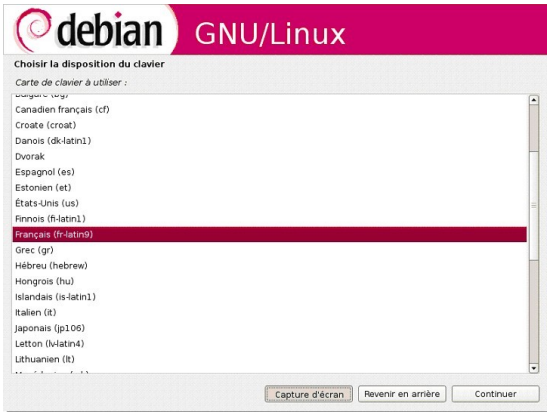
Sélectionner *fr_FR@euro* et appuyer sur *Continuer*.

S'assurer que rien n'est coché et appuyer sur *Continuer*.

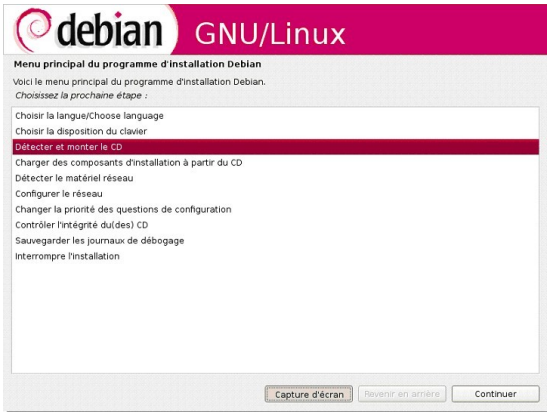


Appuyer sur *Continuer*.

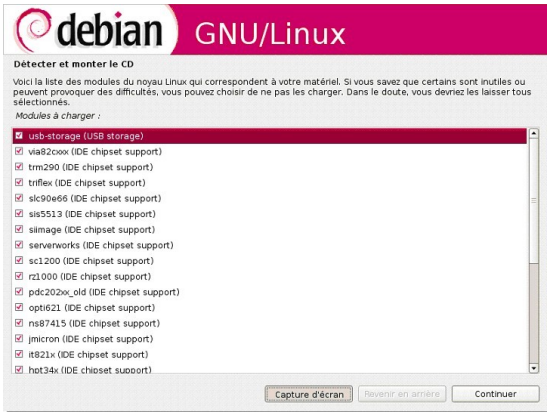
Appuyer sur *Continuer*.



Appuyer sur *Continuer*.



Appuyer sur *Continuer*.



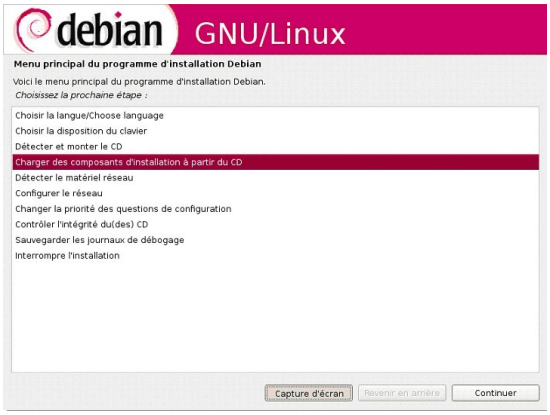
Sauf si vous savez ce que vous faites, laisser tout coché et appuyer sur *Continuer*.

… Détection du matériel …

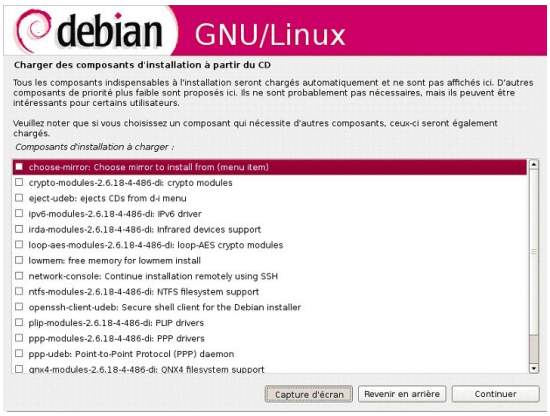
Sélectionner *Non* et appuyer sur *Continuer*.

… Examen du CD …

Appuyer sur *Continuer*.

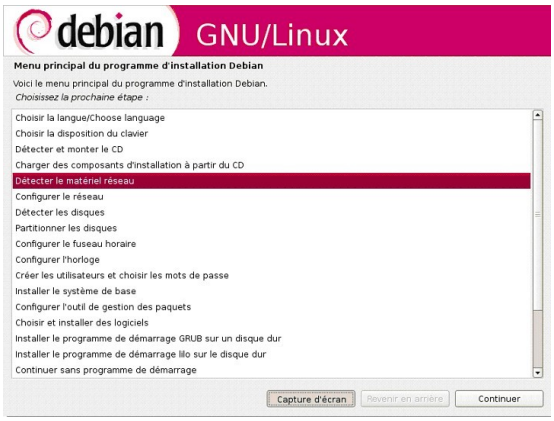


Appuyer sur *Continuer*.



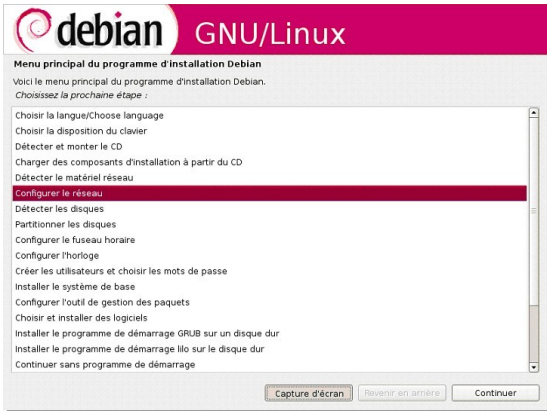
Sauf s'il l'on veut monter du RAID (module mdcfg) ou autre spécificité, ne rien sélectionner et appuyer sur *Continuer*.

… Charger des composants supplémentaires …



Appuyer sur *Continuer*.

… Détection du matériel réseau …



Appuyer sur *Continuer*.

Cette affichage n'apparaît que si la machine possède plusieurs cartes réseaux.
Sélectionner la carte réseau utilisée pour l'installation et appuyer sur *Continuer*.

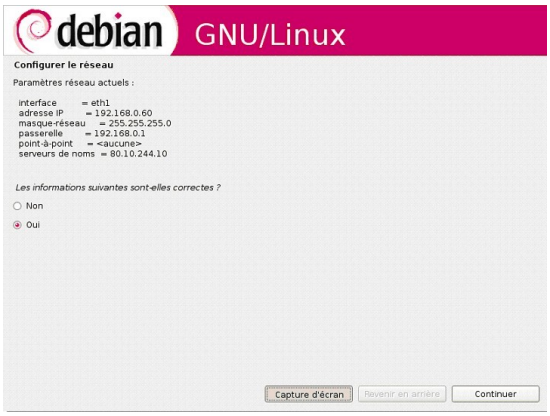
Choisir *Non* et appuyer sur *Continuer*.

Indiquer l'adresse IP de la machine (pour l'interface choisie précédemment) et appuyer sur *Continuer*.

Indiquer le masque sous-réseau et appuyer sur *Continuer*.

Indiquer la passerelle et appuyer sur *Continuer*.

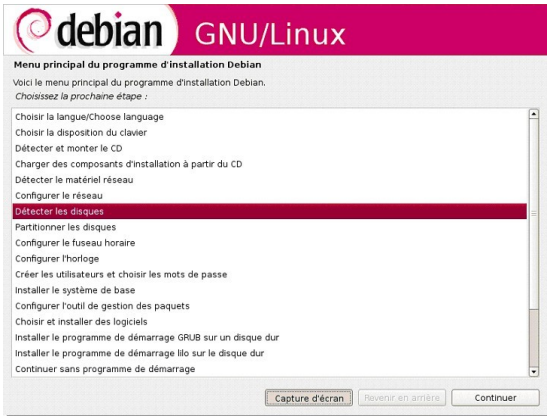
Indiquer les adresses IP des serveurs de nom et appuyer sur *Continuer*.



Si le récapitulatif proposé est correct, sélectionner *Oui* et appuyer sur *Continuer*.

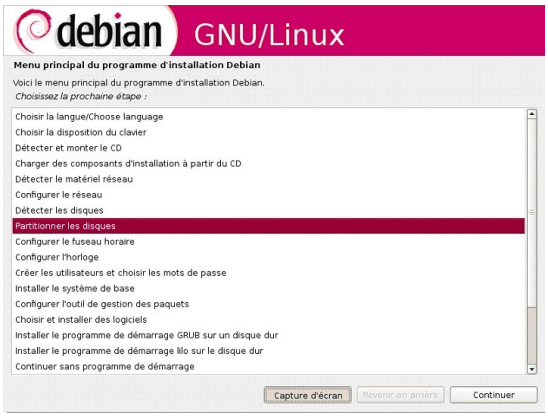
Indiquer le nom de la machine (ici *Cobaye*) et appuyer sur *Continuer*.

Laisser vide le domaine et appuyer sur *Continuer*.



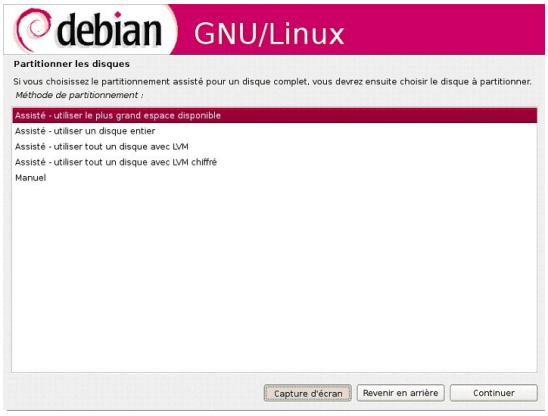
Appuyer sur *Continuer*.

… Détecter les disques …



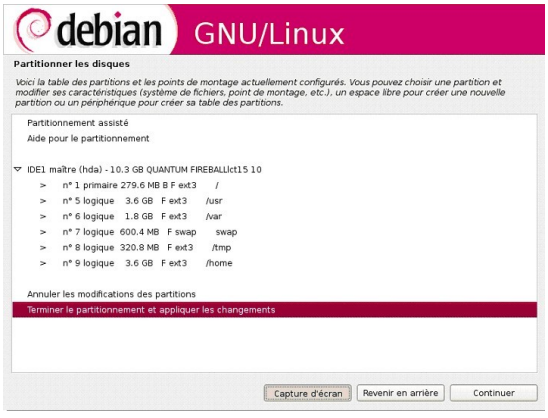
Appuyer sur *Continuer*.

… Démarrage de l'outil de partitionnement …

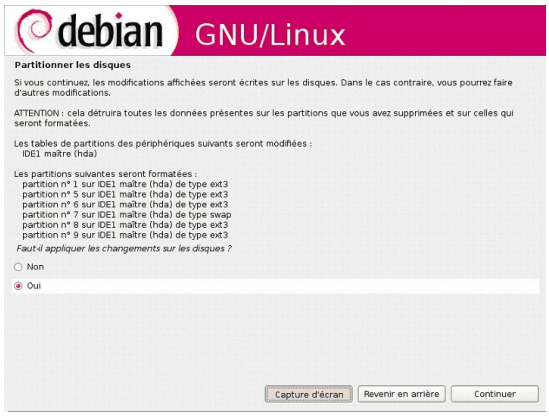


Il est préférable de partitionner soit-même le disque dur (*Manuel*), mais si l'on ne sait pas faire, utiliser l'assistant (c'est le choix décrit dans cette documentation) :
Sélectionner *Assisté – utiliser le plus grand espace disponible* puis appuyer sur *Continuer*.

Sélectionner *Partitions /home, /usr, /var et /tmp séparées* puis appuyer sur *Continuer*.

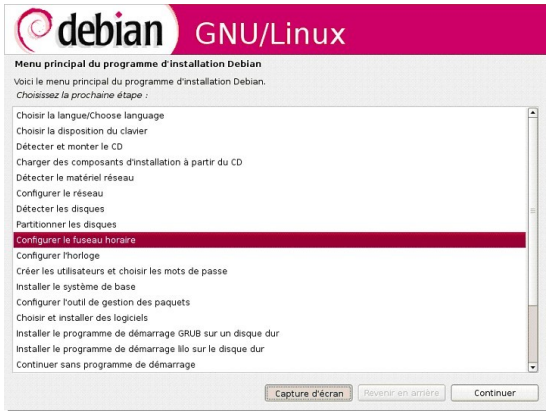


Sélectionner *Terminer le partitionnement et appliquer les changements* puis sélectionner *Continuer*.



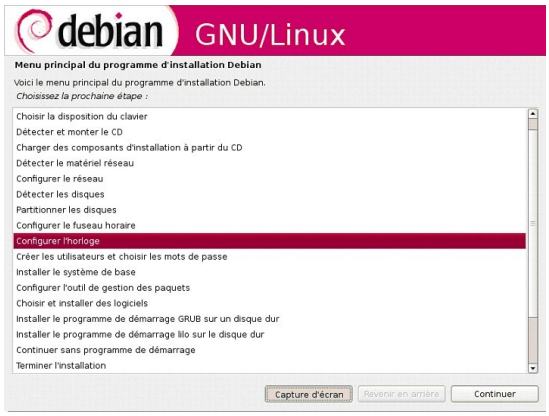
Si le récapitulatif nous convient appliquer-les en sélectionnant *Oui* puis en appuyant sur *Continuer*.

… Formatage des partitions …



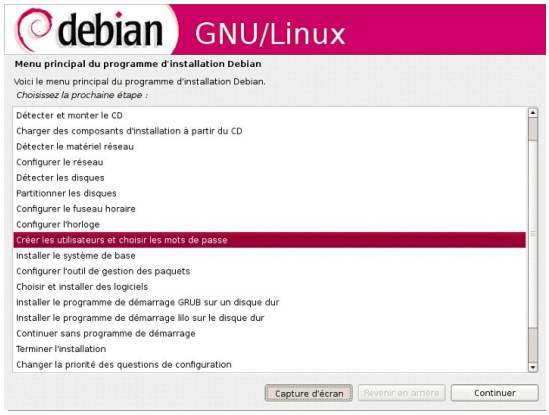
Appuyer sur *Continuer*.

Appuyer sur *Continuer*.



Appuyer sur *Continuer*.

Sélectionner *Oui* puis appuyer sur *Continuer*.



Appuyer sur *Continuer*.

Choisir *Oui* dans les deux cas et appuyer sur *Continuer*.

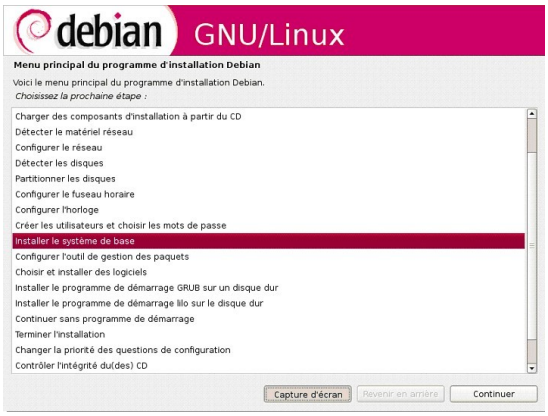
Rentrer le mot de passe du root dans le premier champ (des étoiles s'affichent) et le rentrer à nouveau dans le second champ. Ensuite appuyer sur *Continuer*.

Choisir *Oui* et appuyer sur *Continuer*.

Dans le champ rentrer *adminRADIUS* puis appuyer sur *Continuer*.

Appuyer sur *Continuer*.

Rentrer le mot de passe de l'utilisateur adminRADIUS dans le premier champ (des étoiles s'affichent) et le rentrer à nouveau dans le second champ. Ensuite appuyer sur *Continuer*.

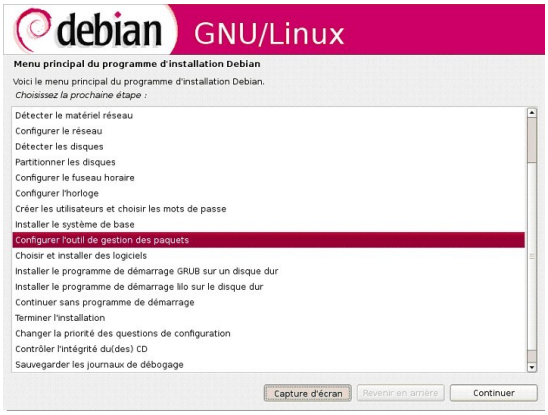


Appuyer sur *Continuer*.

… Installation du système de base … *(3,15 min d'attente environ)*

Sauf CPU ancienne, sélectionner *linux-image-2.6-686* puis appuyer sur *Continuer*.

… Installation du système de base … *(1,10 min d'attente environ)*

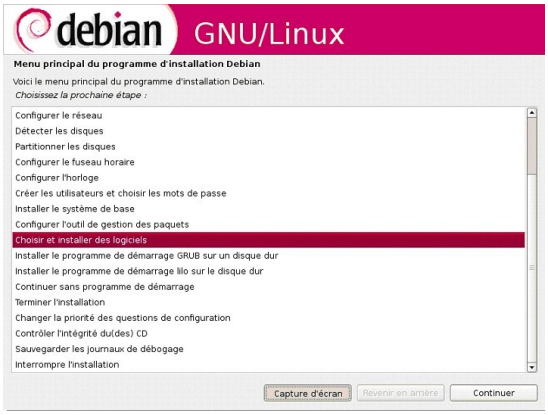


Appuyer sur *Continuer*.

Choisir *Non* et appuyer sur *Continuer*.

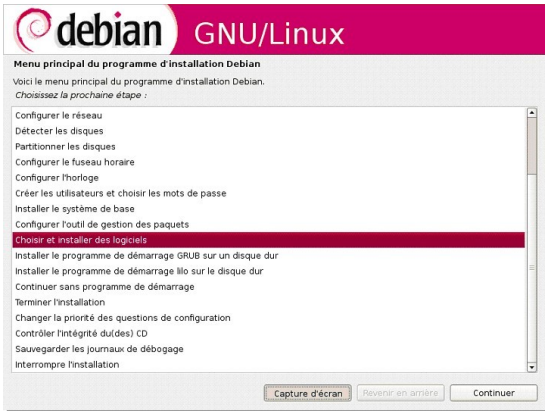
… Configuration de l'outil de gestion des paquets (APT) …

Il est fortement probable que nous ayons ce message d'erreur si le PC ne peut pas se connecter à Internet. Ce message nous indique que le serveur a essayé de contacter un autre serveur sur Internet pour faire une mise à jour de sécurité et qu'il n'y est pas arrivé. Appuyer sur *Continuer*.



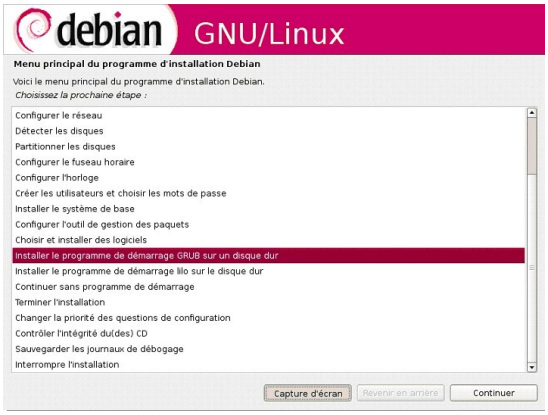
Appuyer sur *Continuer*.

Choisir *Non* et appuyer sur *Continuer*.



Appuyer sur *Continuer*.

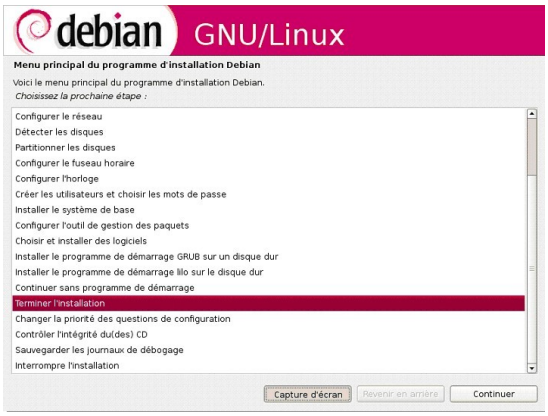
Tout décocher puis appuyer sur *Continuer*.



Appuyer sur *Continuer*.

Choisir *Oui* puis appuyer sur *Continuer*.

Laisser vide le champ et appuyer sur *Continuer*.



Appuyer sur *Continuer*.

… Fin de l'installation …

Enlever le cédérom du lecteur (qui s'éjecte tout seul) et appuyer sur *Continuer*.

La machine redémarre pour nous proposer un prompt de connexion. C'est terminé.

Fichiers créés par l'installateur

Dans le fichier */var/log/installer/lsb-release*, on voit quelle est la version de l'installateur. Ci-dessous un exemple :

```
DISTRIB_ID=Debian
DISTRIB_DESCRIPTION="Debian GNU/Linux installer"
DISTRIB_RELEASE="4.0 (installer build 20070308)"
X_INSTALLATION_MEDIUM=cdrom
```

On peut voir la configuration matérielle de la machine, les modules chargés, et pleins d'autres choses encore dans le fichier */var/log/installer/hardware-summary* .

On peut contrôler le partitionnement réalisé dans le fichier */var/log/installer/partman* .

Le fichier */var/log/installer/status* indique les logiciels installés.

Les logs de l'installation sont accessibles dans le fichier */var/log/installer/syslog* .

Mise à jour du système

Pour mettre à jour le système il est nécessaire de se connecter à un dépôt de paquet GNU/Linux Debian se trouvant sur le réseau de l'entreprise ou sur Internet (en passant par un proxy par exemple), puis d'obtenir une liste à jour des paquets et enfin de mettre à jour le système ou d'installer le logiciel voulu.
Mais tout d'abord, nous allons configurer *apt* pour qu'il aille dorénavant chercher les paquets sur le dépôt et non plus sur le CD-ROM. Pour cela éditer (par exemple avec *nano*) le fichier */etc/apt/sources.list* pour y rajouter :

```
deb http://debian.der.XXXX.fr/debian etch main contrib
deb-src http://debian.der.XXXX.fr/debian etch main contrib
```

et commenter les lignes commençant par :

```
deb cdrom
```

Pour aller chercher les mises-à-jour de sécurité, décommenter les lignes (attention, cela demande d'avoir un accès à Internet. Voir ci-dessous comment se connecter à Internet à travers un proxy) :

```
deb http://security.debian.org/ etch/updates main contrib
deb-src http://security.debian.org/ etch/updates main contrib
```

Fin de la modification du fichier */etc/apt/sources.list*.

Pour aller chercher des paquets sur Internet il peut être nécessaire de passer par un proxy. Voici donc la configuration de la connexion à Internet à travers le proxy où MonLogin et MonMotDePasse sont à remplacer par les paramètres d'un compte Internet valide. Cela se configure en console :

```
http_proxy=http://MonLogin:MonMotDePasse@www-config-proxy.domaine.com:3128
export http_proxy
```

et pour vérifier que la configuration du proxy est bien prise en compte :

```
echo $http_proxy
```

qui doit retourner

```
http://MonLogin:MonMotDePasse@www-config-proxy.domaine.com:3128
```

Explication de la ligne de commande

```
http_proxy=http://MonLogin:MonMotDePasse@www-config-proxy.domaine.com:3128
```

Cette commande déclare une chaîne de caractère comme valeur de la variable *http_proxy*. On déclare le protocole utilisé (*http*) puis le login connu par le proxy (*MonLogin*) et le mot de passe associé (*MonMotDePasse*). Enfin, on donne l'adresse IP (de préférence) ou le nom DNS complet du proxy http (*www-config-proxy.domaine.com*). Pour terminer on indique le port de connexion au proxy http (*3128*).

Il n'est pas toujours facile de trouver l'adresse IP ou le nom DNS du proxy (*www-config-proxy.domaine.com*). On peut retrouver cette information en cherchant sur un PC duquel on peut se connecter à Internet. Il faut chercher dans les menus de configuration de la connexion du navigateur Internet. Il se peut que seul le nom (ou l'adresse IP) du serveur soit indiquée. Mais il est possible qu'un nom de fichier de configuration soit rajouté. Dans ce dernier cas, la ligne ne se présente pas sous la forme *www-config-proxy.domaine.com*, mais plutôt *www-config-*

proxy.domaine.com/proxy-FichierConf.pac. Dans notre cas nous n'utilisons pas ce fichier de configuration. Ainsi, il ne faut bien évidemment inclure dans la ligne de commande que *www-config-proxy.domaine.com* !

Obtention de la liste à jour des paquets :
```
apt-get update
```

Mise-à-jour du système :
```
apt-get upgrade
```

Penser à clore la session, une fois terminé, afin de détruire la variable *$http_proxy*.

Suppression des logiciels inutiles

Pour supprimer les logiciels et bibliothèques inutiles installés par défaut, taper la commande suivante (il est possible que certains programmes soient réinstallés dans la suite du document) :
```
apt-get --purge remove aptitude dhcp3-client dhcp3-common dmidecode dselect ed eject
installation-report laptop-detect libsasl2 nano tasksel tasksel-data vim-common vim-tiny wget
```

Installation des logiciels supplémentaires

Nous détaillerons l'installation, et plus loin la configuration, d'un watchdog logiciel (*watchdog*), d'un client et serveur ntp (*openntpd*), d'un serveur syslog (*syslogd* déjà installé par défaut), d'un client/serveur ssh (*openssh-client* et *openssh-server*), d'un système de gestion de bases de données (*mysql-server*) et d'un serveur RADIUS (*freeradius* et *freeradius-mysql*). On rajoutera aussi la fonctionnalité de signature des paquets (*debian-keyring* **et** *debian-archive-keyring***).** Pour configurer la carte ethernet, nous installerons également *ethtool*.
Pour la console d'administration il est nécessaire d'installer des bibliothèques perl (*libdbi-perl* et *libdate-calc-perl*). Nous utiliserons la coloration syntaxique des journaux avec *lwatch* (lié à *libpcre3*).

Pour l'installation des paquets manquants, taper la commande suivante :
```
apt-get install openntpd openssh-client openssh-server freeradius freeradius-mysql mysql-
server libdbi-perl libdate-calc-perl debian-keyring debian-archive-keyring ethtool lwatch
libpcre3
```

Sauvegarde/Restauration de la liste des logiciels à installer

Sauvegarder la liste des logiciels installés :
```
dpkg --get-selections |gzip > deb-selections.gz
```

NOTE

Si plus tard nous voulons mettre à jour la liste des paquets à installer/supprimer en utilisant la liste récupérée suite à la commande ci-dessus, il suffira de taper les commandes suivantes :
```
zcat deb-selections.gz | dpkg --set-selections
```
Puis pour lancer la mise-à-jour du système avec la nouvelle liste des logiciels :
```
apt-get update && apt-get upgrade
```

Nettoyage des comptes utilisateurs

L'installation du système crée de nombreux utilisateurs et groupes inutiles que nous allons maintenant supprimer :
```
userdel games
userdel lp
userdel mail
userdel news
userdel uucp
userdel proxy
userdel www-data
userdel list
userdel irc
```
`userdel nobody` (demander confirmation)

Configuration des comptes utilisateurs systèmes (uniquement sur le serveur principal)

Attention, cette rubrique permet de créer les utilisateurs normaux, pas le compte administrateur du système ! Ce dernier sera configuré plus en avant lors de l'installation de la console d'administration.

Créer le groupe *RADIUSgroup* avec la commande suivante :
```
groupadd RADIUSgroup
```
Créer les répertoires */adminradius* et */home/RADIUSuser* :
```
mkdir /adminradius
mkdir /home/RADIUSuser
chown –R 1001.1001 /home/RADIUSuser
```
Copier les fichiers *ChgtMdP.pl* et *chroot* dans le répertoire */adminradius*. Ensuite leur donner les droits adéquats :
```
chmod 755 /adminradius/ChgtMdP.pl
chmod 755 /adminradius/chroot
```
et s'assurer du propriétaire des fichiers :
```
chown adminradius.adminradius /adminradius/ChgtMdP.pl
chown adminradius.adminradius /adminradius/chroot
```
Ensuite pour créer les utilisateurs, nous utiliserons un script qui créera à la fois les utilisateurs dans FreeRADIUS et dans le système. Les commandes sont dans le programme *netadmin.pl* (se reporter à la section présentant la console d'administration).

Quelques explications sont nécessaires pour comprendre pourquoi nous utilisons tous ces scripts. Dans la console d'administration, en même temps que l'on configure le profil de l'utilisateur sous FreeRADIUS, on définit aussi son profil système. Biensûr, avant de lui donner un mot de passe dans le système on lui a créé un compte. Les caractéristiques de ce compte sont :

- Le HOME est */adminradius* (comme pour tous les utilisateurs normaux)
- Le shell est */adminradius/chroot*. C'est en fait un script qui appelle l'interpréteur perl pour exécuter un script perl nommé *ChgtMdP.pl*. Ce script guide l'utilisateur dans le changement de son mot de passe dans FreeRADIUS et dans le système. Ainsi, lorsque un utilisateur se connecte au système, il n'a pas de shell classique mais lance automatiquement le script perl pour changer le mot de passe et dès que le script est terminé, l'utilisateur est automatiquement déconnecté.
- Le groupe est *RADIUSgroup*. La commande utilisée dans le script pour créer l'utilisateur système (commande *adduser*) ne crée pas le groupe s'il n'existe pas et s'interrompt avec une erreur. C'est pour cela que l'on crée le groupe (`groupadd RADIUSgroup`) avant la création du moindre utilisateur.

NOTE

Comme les mots de passes système et FreeRADIUS se configurent séparément, il est possible de choisir des mots de passes différents. C'est possible mais non souhaitable pour des raisons de facilité de mémorisation.

Lorsque l'utilisateur change son mot de passe, il change son mot de passe système puis son mot de passe FreeRADIUS. Il peut alors arriver que le mot de passe soit changé pour le système et qu'il y ait un échec avec le mot de passe FreeRADIUS ou l'inverse. Généralement c'est parce que le mot de passe est trop court ou trop simple ou que l'on n'a pas rentré le bon mot de passe courant ou encore que l'on n'a pas confirmé correctement le nouveau mot de passe. Pour résoudre ce problème il faut se reconnecter avec l'ancien mot de passe (si seul le mot de passe FreeRADIUS a été changé) ou avec le nouveau (si seul le mot de passe système a été changé) et recommencer la procédure.

Configuration de netfilter

Utiliser le script de configuration du firewall, disponible en annexe (attention, selon que l'on se trouve sur le serveur principal ou sur le serveur secondaire, le script à reprendre n'est pas le même). Remplacer $SERVEUR (ou l'adresse IP indiquée pour exemple dans le script) par l'adresse IP du serveur qui héberge ce script et indiquer la bonne interface réseau (eth0 ou eth1, vu pendant l'installation). Le nommer *XXXXFirewall* en le copiant dans */etc/init.d/* puis établir le lien symbolique :
```
mv <chemin actuel script> /etc/init.d/XXXXFirewall
ln –s  /etc/init.d/XXXXFirewall   /etc/rc2.d/S98ScriptXXXXFirewall
```

Le script *XXXXFirewall* doit avoir les mêmes droits et les mêmes propriétaires que les autres scripts du répertoire */etc/init.d/* :
```
cd /etc/init.d
chown root.root XXXXFirewall
chmod 750 XXXXFirewall
```

Configuration du watchdog logiciel

Pour activer le watchdog au démarrage, décommenter dans le script du firewall la ligne *modprobe softdog*. Puis rajouter dans le fichier */etc/crontab* la ligne suivante :

```
* * * * * adminradius sleep 30 && /adminradius/netadmin.pl –A –choix adm –sschoix watch
```
Attention, si pour des raisons de maintenance nous sommes amenés à arrêter MySQL ou FreeRADIUS ou syslogd pour plus d'une minute, décharger le module *softdog* (commande *rmmod softdog*) ou arrêtez *crond* (commande */etc/init.d/cron stop*) pour éviter un redémarrage intempestif.

Installer un éditeur de texte

Pour la suite, il sera nécessaire d'éditer des fichiers. 2 éditeurs (parmi beaucoup d'autres) très légers sont *nano* et *jed* (ce dernier est d'utilisation proche d'*emacs*). Pour installer *jed* (même chose pour *nano*), taper la commande (il sera nécessaire de déclarer à nouveau les paramètres pour se connecter au proxy http, cf chapitre « Mise à jour du système ») :
```
apt-get install jed
```

Configuration de /etc/fstab

On autorise des personnes à se connecter au système pour changer leur mot de passe. Il va donc falloir s'assurer qu'elles ne puissent pas utiliser le système comme serveur de fichier ou pour exécuter des fichiers. Pour cela on configure */etc/fstab* avec l'option *noexec, ro* pour le */home*. C'est-à-dire que l'on rajoute (indiqué en gras) la ligne ressemblant à :
```
/dev/hda9     /home     ext3    defaults    0    2
```
par
```
/dev/hda9     /home     ext3    defaults,noexec,ro    0    2
```

Configuration de openntpd

Pour avoir un serveur NTP sur la machine, nous avons installé *openntpd*. Il sert à la fois de client et de serveur NTP. Maintenant pour le configurer il suffit de renseigner dans le fichier */etc/openntpd/ntpd.conf* l'adresse IP sur laquelle le serveur de temps répond et le serveur externe (192.168.126.254 par exemple) sur lequel on se synchronise :
```
listen on <@IP Serveur> (commenter cette ligne sur le serveur secondaire)
server <@IP du serveur ntp externe>
```

Toutes les autres lignes doivent être commentées.

Puis nous allons réaliser une mise à l'heure « brutale » du système à chaque démarrage du daemon (avec la manière « douce », il peut être nécessaire d'attendre la mise à l'heure plusieurs heures voir plusieurs jours si le décalage est de plusieurs minutes voir de plusieurs heures). Pour cela, dans le fichier */etc/default/openntpd* décommenter :
```
DAEMON_OPTS="-s"
```

Puis redémarrer le serveur :
```
/etc/init.d/openntpd restart
```

Configuration de syslog (uniquement sur le serveur principal)

Dans le fichier */etc/default/syslogd*, remplacer
```
SYSLOGD=""
```
par
```
SYSLOGD="-r"
```

Dans le fichier `/etc/syslog.conf`, décommenter et modifier la ligne
```
#cron.*                         /var/log/cron.log
```
par
```
cron.*                         -/var/log/cron.log
```

On peut vouloir faire défiler les logs en temps-réel sur les consoles sans devoir s'y logguer. Pour cela décommenter les lignes suivantes dans le script du firewall :
```
killall lwatch
killall cat
killall tail
cat > /dev/xconsole | lwatch –i - >> /dev/tty8 &
tail –f /var/log/freeradius/radius.log | lwatch –i - >> /dev/tty9 &
```

La première ligne affiche le contenu de la console virtuelle *xconsole* (donc les logs systèmes) dans la console 8. *lwatch* est utilisé pour la colorisation syntaxique.

La seconde ligne affiche les modifications dans le fichier de log radius, dans la console 9. *lwatch* est une nouvelle fois utilisé pour la coloration synthaxique.

Gestion du stockage des logs

Les journaux systèmes sont stockés et archivés, certains par défaut trop peu de temps pour aider à retrouver des informations de plusieurs semaines dans le passé. Nous allons reconfigurer *logrotate* pour qu'il réponde à nos besoins. Nous allons demander que par défaut les fichiers soient sauvegardés 52 semaines et les archives compressées (en réalité le fichier de la semaine en cours ainsi que de la semaine précédente ne sont pas compressés). Dans le fichier */etc/logrotate.conf*, s'assurer de retrouver :

```
weekly
```

Et remplacer

```
rotate 4
```

par

```
rotate 52
```

Puis décommenter

```
compress
```

Nous venons de définir les paramètres par défaut. Nous allons personnaliser les paramètres spécifiquement pour les journaux FreeRADIUS et Syslog. Pour cela, dans le répertoire */etc/logrotate.d/*, créer les fichiers *syslog* et *freeradius* avec les bons propriétaires et les bons droits :

```
touch /etc/logrotate.d/syslog
chmod 644 /etc/logrotate.d/syslog
chown root.root /etc/logrotate.d/syslog
touch /etc/logrotate.d/freeradius
chmod 644 /etc/logrotate.d/freeradius
chown root.root /etc/logrotate.d/freeradius
```

Puis écrire dans le fichier *syslog* :

```
/var/log/syslog {
       weekly
       rotate 52
       compress
       notifempty
}
```

et dans le fichier *freeradius* :

```
/var/log/freeradius/*.log {
        weekly
        rotate 52
        compress
        notifempty
}
```

Configuration de MySQL pour FreeRADIUS

S'assurer la présence dans le fichier */etc/mysql/my.cnf* des lignes (et des bonnes valeurs) :

```
port = 3306          (cette ligne apparaît 2 fois dans le fichier)
bind-address = 127.0.0.1
```

Maintenant nous allons définir les bases de données et leurs tables.
Soit *db_mysql.sql* le script que nous allons utiliser. Ce script est disponible en annexe. Il est inspiré du script disponible sur http://wiki.freeradius.org/MySQL_DDL_script auquel on a fait quelques modifications. Ce script crée la base de données et les tables. Il configure aussi les logins et mots de passe pour accéder au SGBD ainsi que les droits associés. La définition des logins, mots de passe et droits associés sont définis dans la dernière rubrique du script. Avant d'exécuter le script, il faut remplacer *MotDePasseRoot* par le mot de passe choisi pour l'utilisateur root (le mot de passe peut être différent de celui utilisé par root sur le système). Remplacer *radiusXXX* par le login de l'utilisateur qui sera utilisé pour gérer et accéder à la base de données et *radiusXXXMdP* par son mot de passe. Ces changements doivent être reportés sur le fichier *AccesBD.txt*.

Pour finir, charger le script. Pour cela se positionner dans le répertoire où se trouve le script et taper la commande suivante :

```
mysql –u root -p < db_mysql.sql
```

Configuration de FreeRADIUS

Tous les fichiers de configuration que nous allons manipuler dans ce chapitre sont dans */etc/freeradius/*.
Dans le fichier *sql.conf* :
S'assurer qu'est décommenté :
`driver = "rlm_sql_mysql"`
et sont commentés les autres drivers.

Donner ensuite les infos pour se connecter à la base de données :
`server = "localhost"`
`login = "radiusXXX"`
`password = "radiusXXXMdP"` (remplacer par le mot de passe défini lors de la création de la base de données)
`radius_db = "radius"`
`readclient = yes` ou `readclients = yes` (cette ligne se trouve en fin de fichier)

Commenter les lignes ci-dessous

```
# The default queries are case insensitive. (for compatibility with
# older versions of FreeRADIUS)
#     authorize_check_query = "SELECT id, UserName, Attribute, Value, op \
#          FROM ${authcheck_table} \
#          WHERE Username = '%{SQL-User-Name}' \
#          ORDER BY id"
#
#authorize_reply_query = "SELECT id, UserName, Attribute, Value, op \
#          FROM ${authreply_table} \
#          WHERE Username = '%{SQL-User-Name}' \
#          ORDER BY id"
```

Rajouter

```
authorize_check_query = "SELECT ${authcheck_table}.id, ${authcheck_table}.UserName, $
{authcheck_table}.Attribute, ${authcheck_table}.Value, ${authcheck_table}.op FROM $
{authcheck_table}, ${usergroup_table}, ${nas_table} WHERE ${authcheck_table}.UserName= BINARY
'%{SQL-User-Name}' AND ${nas_table}.groupe=${usergroup_table}.GroupName AND $
{usergroup_table}.UserName=${authcheck_table}.UserName ;"
```

Décommenter les lignes suivantes :

```
authorize_reply_query = "SELECT id, UserName, Attribute, Value, op \
          FROM ${authreply_table} \
          WHERE UserName = BINARY '%{SQL-User-Name}' \
          ORDER BY id ;"
```

Décommenter les lignes ci-dessous :

```
# Use these for case sensitive usernames.
      authorize_group_check_query = "SELECT ${groupcheck_table}.id,$
{groupcheck_table}.GroupName,${groupcheck_table}.Attribute,${groupcheck_table}.Value,$
{groupcheck_table}.op FROM ${groupcheck_table},${usergroup_table} WHERE $
{usergroup_table}.Username = BINARY '%{SQL-User-Name}' AND ${usergroup_table}.GroupName = $
{groupcheck_table}.GroupName ORDER BY ${groupcheck_table}.id"
      authorize_group_reply_query = "SELECT ${groupreply_table}.id,$
{groupreply_table}.GroupName,${groupreply_table}.Attribute,${groupreply_table}.Value,$
{groupreply_table}.op  FROM ${groupreply_table},${usergroup_table} WHERE $
{usergroup_table}.Username = BINARY '%{SQL-User-Name}' AND ${usergroup_table}.GroupName = $
{groupreply_table}.GroupName ORDER BY ${groupreply_table}.id"
```

et commenter les lignes suivantes :

```
#     authorize_group_check_query = "SELECT ${groupcheck_table}.id,$
{groupcheck_table}.GroupName,${groupcheck_table}.Attribute,${groupcheck_table}.Value,$
{groupcheck_table}.op  FROM ${groupcheck_table},${usergroup_table} WHERE $
{usergroup_table}.Username = '%{SQL-User-Name}' AND ${usergroup_table}.GroupName = $
{groupcheck_table}.GroupName ORDER BY ${groupcheck_table}.id"
#     authorize_group_reply_query = "SELECT ${groupreply_table}.id,$
{groupreply_table}.GroupName,${groupreply_table}.Attribute,${groupreply_table}.Value,$
{groupreply_table}.op  FROM ${groupreply_table},${usergroup_table} WHERE $
{usergroup_table}.Username = '%{SQL-User-Name}' AND ${usergroup_table}.GroupName = $
{groupreply_table}.GroupName ORDER BY ${groupreply_table}.id"
```

Dans le fichier *radiusd.conf* :
Changer et décommenter
```
bind-address = *
port = 0
proxy_requests = yes
```
en
```
bind-address = <@IP serveur>
port = 1812
proxy_requests = no
```
et
```
log_auth = no
log_auth_badpass = no
```
en
```
log_auth = yes
log_auth_badpass = yes
```

Décommenter
```
detail auth_log {
                detailfile = ${radacctdir}/%{Client-IP-Address}/auth-detail-%Y%m%d
                detailperm = 0600
}
```

Puis dans la section *authorize*,
commenter
```
# mschap
# eap
# files
```
et décommenter
```
auth_log
sql
```

Dans la section *authenticate*,
commenter
```
# unix
# eap
```

Dans la section *accounting*,
décommenter
```
sql
```

Dans la section *post-auth*,
décommenter
```
sql
```

Pour que tous ces changements soient pris en compte, redémarrer FreeRADIUS :
```
/etc/init.d/freeradius restart
```

NOTE :

Signification des tables :
nas : remplace le fichier client.conf. On y stocke les nas.
radacct : on y stocke les informations que le nas retourne pour tout ce qui accounting
radcheck : cette table contient les attributs à vérifier lors d'une authentification
radgroupcheck : idem pour des groupes
radreply : cette table contient les attributs et les valeurs associées, à renvoyer à l'utilisateur (on ne renvoie que la première)
radgroupreply : idem pour des groupes (c'est ici que l'on met 3Com-User-Access-Level=3, ect)
radpostauth : les informations de chaque authentification réussie y sont stockées.
usergroup : cette table permet de faire la correspondance entre le nom d'utilisateur et son groupe.

*Configuration de openssh-**

Configuration du client (seulement sur le serveur principal) :
Ne rien faire.

Configuration du serveur :
Cette configuration s'effectue dans le fichier */etc/ssh/sshd_config*. Le modèle est ci-dessous. Y remplacer 192.168.0.44 par l'adresse IP du serveur. Les lignes commençant par # indiquent les valeurs par défaut que l'on ne modifie pas. Les lignes commençant par ## indiquent les options non-utilisées :

```
# AddressFamily any
AllowGroups
# AllowTcpForwarding yes
AllowUsers
AuthorizedKeysFile ./ssh/authorized_keys
# Banner monTexte.txt
# ChallengeResponseAuthentication yes
# Ciphers aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour128,arcfour256,arcfour,aes192-cbc,aes256-cbc,aes128-ctr,aes192-ctr,aes256-ctr
ClientAliveCountMax 4
# ClientAliveInterval 15
Compression no
## DenyGroups
## DenyUsers
## ForceCommande
# GatewayPorts no
# GSSAPIAuthentication no
# GSSAPICleanupCredentials yes
# HostbasedAuthentication no
# HostbasedUsesNameFromPacketOnly no
## Hostkeys /etc/ssh/ssh_host_rsa_key
## Hostkeys /etc/ssh/ssh_host_dsa_key
# IgnoreRhosts yes
# IgnoreUserKnownHosts no
# KerberosAuthentication no
# KerberosGetAFSToken no
KerberosOrLocalPasswd no
# KeyRegenerationInterval 3600
ListenAddress 192.168.0.44
# LoginGraceTime 120
# LogLevel INFO
# MACs hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160,hmac-sha1-96,hmac-md5-96
## Match
# MaxAuthTries 6
# MaxStartups 10
# PasswordAuthentication yes
# PermitEmptyPasswords no
## PermitOpen
PermitRootLogin yes # mettre “no” si l’on veut interdire les connexions root directes.
# PermitTunnel no
# PermitUserEnvironment no
# PidFile /var/run/sshd.pid
# Port 22
# PrintLastLog yes
PrintMotd no
Protocol 2
# PubkeyAuthentication yes
## RhostsRSAAuthentication no
## RSAAuthentication yes
## ServerKeyBits 768
# StrictModes yes
## Subsystem
# SyslogFacility AUTH
# TCPKeepAlive yes
UseDNS no
# UseLogin no
# UsePrivilegeSeparation yes
## X11DisplayOffset 10
# X11Forwarding no
## X11UseLocalhost yes
```

```
## XauthLocation /usr/X11R6/bin/xauth
```

Configuration de sshfs

(donné pour information ; non-utilisé dans notre architecture)

Avant chaque utilisation de sshfs, il faudra activer le module *fuse*. Cela peut-être réalisé lors du démarrage de la machine en le mettant par exemple dans le script du firewall (cf script firewall plus bas). S'il est nécessaire de le faire manuellement, taper la commande suivante en tant que root :

```
modprobe fuse
```

Pour configurer sshfs :

- Créer sur chaque serveur un utilisateur (*vpnssh* dans le groupe *fuse*) qui sera utilisé uniquement pour monter le système de fichier : `useradd vpnssh fuse`
- Donner un mot de passe à vpnssh : `passwd vpnssh`
- Créer les répertoires à partager sur les 2 serveurs :
- Sur le serveur principal : `mkdir /home/vpnssh/Principal && mkdir \ /home/vpnssh/SecoursSurPrincipal`
- Sur le serveur de secours : `mkdir /home/vpnssh/Secours`
- Attention, l'utilisateur vpnssh ne peut monter un système de fichiers distant que dans un répertoire local dans lequel il a un accès en écriture.
- Ensuite monter le système de fichier sur le serveur principal avec la commande suivante (peut être insérée dans un script exécuté au démarrage de la machine) :
  `sshfs vpnssh@serveurSecours:/home/vpnssh/Secours /home/vpnssh/SecoursSurPrincipal`
- Ainsi, tout fichier écrit dans *serveurSecours:/home/vpnssh/Secours* sera disponible sur *seveurPrincipal:/home/vpnssh/SecoursSurPrincipal* et inversement, comme si c'était le même répertoire.
- Pour démonter le système de fichier distant depuis le serveur principal, il suffit de taper la commande :
  `fusermount –u /home/vpnssh/SecoursSurPrincipal`

Configuration d'un tunnel ssh

Nous allons monter un VPN SSH entre le serveur principal et le serveur secondaire. Il faut donc que le serveur de secours soit déjà existant pour faire ce qui suit. Ce VPN sera utilisé pour synchroniser la base de données du serveur secondaire depuis le serveur principal, et cela en faisant transiter les données par ce tunnel chiffré. On utilise l'utilisateur vpnssh. S'il n'a pas encore été créé, le faire sur le serveur de secours où l'on tapera la commande :

```
useradd vpnssh
```

et on lui affectera un mot de passe :

```
passwd vpnssh
```

Explication de la ligne de commande

Pour la machine sur laquelle on a tapé la commande ci-dessous, tous les flux qui lui parviennent sur le port 30000 seront redirigés par le tunnel ssh (port 22) sur le serveur 192.168.0.44. Ensuite on demande au serveur 192.168.0.44 de rediriger tous les flux qui lui parviennent par le tunnel sur le port 3306 de la machine ayant pour adresse 127.0.0.1. Ainsi, on peut accéder à la base MySQL du serveur 192.168.0.44, qui n'accepte pourtant que les connexions locales. Elle est donc sécurisée contre les attaques extérieures tout en étant accessible depuis l'autre serveur. Si la base était accessible depuis l'extérieur (donc dans le fichier */etc/mysql/my.cnf*, *bind-address = 192.168.0.44*), on aurait mis 192.168.0.44 à la place de 127.0.0.1 dans la commande ci-dessous.
Une autre possibilité (moins intéressante dans le fonctionnement que l'on a choisi) aurait été de demander à la machine distante de monter un tunnel ; elle aurait donc du utiliser l'option –R à la place de –L.

Pour monter le VPN, taper la commande suivante (remplacer 192.168.0.44 par l'adresse du serveur de secours) :

```
ssh –N –f vpnssh@192.168.0.44 –L 30000:127.0.0.1:3306
```

Au montage du VPN, le mot de passe de l'utilisateur vpnssh déclaré sur le serveur de secours sera demandé.
L'option *–f* fait passer le processus en arrière-plan. Pour le tuer, il faudra donc utiliser la commande *kill <pid>* ou plus simplement taper la commande (attention, elle tue tous les processus ssh, même ceux n'ayant rien à voir avec le tunnel comme la connexion d'un utilisateur au serveur) :

```
killall ssh
```

A chaque redémarrage d'un des 2 serveurs, il faudra remonter le tunnel manuellement (il est nécessaire de rentrer un mot de passe).

Pour interdire d'utiliser le login vpnssh pour une connexion standard au système, dans le fichier */etc/passwd*, remplacer
`vpnssh:x:1000:1002::/home/vpnssh:`**`/bin/sh`**
par
`vpnssh:x:1000:1002::/home/vpnssh:`**`/bin/false`**

Configuration de telnetd

Nous allons configurer un serveur telnet sur cette machine. Mais c'est seulement pour faire une transition avec l'ancienne méthode d'administration. A terme ssh devra lui être préféré, et telnet supprimé.
Pour installer le serveur *telnet*, taper la commande suivante :
```
apt-get install telnetd openbsd-inetd
```

Pour le configurer :
Dans le fichier */etc/inetd.conf*, rajouter ou décommenter la ligne suivante :
```
telnet stream  tcp nowait telnetd /usr/sbin/tcpd  /usr/sbin/in.telnetd
```

S'assurer que dans */etc/rc2.d/* le daemon *openbsd-inetd* soit bien lancé.

Ajout de routes supplémentaires

Dans le cas où la passerelle donnée ne permet pas d'accéder à certains réseaux (par exemple le RIH pour le XXX), il est nécessaire de rajouter des routes qui seront intégrées au démarrage de la machine. Pour cela, rajouter au script du firewall les lignes ci-dessous (à adapter). Dans notre cas, seules les 2 premières lignes sont à rajouter sur le serveur principal et seules les 2 dernières sur le serveur de secours :
```
route add –net 172.16.0.0 netmask 255.255.240.0 gw 172.16.69.3
route add –net 172.16.16.0 netmask 255.255.240.0 gw 172.16.69.3

route add –net 172.16.0.0 netmask 255.255.240.0 gw 172.16.80.1
route add –net 172.16.16.0 netmask 255.255.240.0 gw 172.16.80.1
```

Configuration de la supervision

Il n'est pas prévu l'installation d'agent SNMP ou autre du même genre. Cependant il est possible de connaître l'état de la machine en utilisant la console d'administration. Si un problème survient dans les principaux services de la machine (syslog, FreeRADIUS, MySQL) le watchdog redémarre la machine. Ainsi, si l'uptime (temps écoulé depuis le dernier démarrage de la machine) est faible c'est qu'il y a eu un problème. Une exploration des journaux systèmes est alors indispensable. D'autres infos sont aussi données par la console d'administration (infos de routage, des processus en écoute du réseau, ...). Pour terminer, il est possible, toujours depuis la console d'administration, de consulter les tentatives réussies ou échouées de connexions à un nas.

Installation de la console d'administration

Créer le répertoire de l'administrateur si cela n'a pas été fait :
```
mkdir /adminradius
```
Copier dans ce répertoire les fichiers *netadmin.pl* et *AccesBD.txt*.
Ensuite leur donner les droits adéquats :
```
chmod 770 netadmin.pl
chmod 644 AccesBD.txt
```
et s'assurer du propriétaire des fichiers :
```
chown adminradius.adminradius netadmin.pl
chown adminradius.adminradius AccesBD.txt
```
Mettre à jour les informations nécessaires à la connexion à la base de données dans le fichier *AccesBD.txt* (voir l'annexe pour comprendre la structure du fichier).
Par défaut, l'utilisateur adminradius est un utilisateur sans privilège particulier. Pour lui donner les droits adéquats (les droits administrateur système !) et définir son répertoire courant, éditer le fichier */etc/passwd* et changer une ligne ressemblant à :
`adminradius:x:`**`1000`**`:`**`1000`**`:adminradius,,,:`**`/home/adminradius`**`:/bin/bash`
en (les éléments à modifier sont en gras)
`adminradius:x:`**`0`**`:`**`0`**`:adminradius,,,:`**`/adminradius`**`:/bin/bash`
détruire l'ancien répertoire personnel d'adminradius :
```
rm –rf /home/adminradius
```
puis dans le fichier */etc/group*, modifier une ligne ressemblant à :

`RADIUSgroup:x:1000:`
en
`RADIUSgroup:x:0:`

Utilisation de la console d'administration

Pour administrer le système comme FreeRADIUS, il n'est pas nécessaire de passer par cette console d'administration. Cependant, pour aider les administrateurs, l'ensemble des principales fonctionnalités, nécessaires à la bonne administration de l'outil, sont disponibles par l'intermédiaire de cette console d'administration. Pour plusieurs tâches, il est nécessaire d'avoir les droits administrateur système. Pour cette raison, la console d'administration ne sera accessible que par les administrateurs.
La console d'administration est un programme écrit en perl nommé *netadmin.pl* .
Ce programme a précédemment été copié dans le compte de l'administrateur adminradius, et il sera donc chargé en tapant :
`./netadmin.pl` si l'on se trouve dans le répertoire */adminradius* ,
`/adminradius/netadmin.pl` sinon.

Les menus disponibles sont (v1.0) :
0 - Quitter l'application
1 - Gestion des utilisateurs
- 11 - Ajouter un utilisateur
- 12 - Supprimer un utilisateur
- 13 - Modifier le mot de passe d'un utilisateur
- 14 - Ajout d'un attribut à un utilisateur
- 15 - Lister un ou tous les utilisateurs

2 - Gestion des groupes
- 21 - Ajouter un attribut de requête à un groupe
- 22 - Ajouter un attribut de réponse à un groupe
- 23 - Lister les attributs de requête
- 24 - Lister les attributs de réponse
- 25 - Supprimer un attribut de requête à un groupe
- 26 - Supprimer un attribut de réponse à un groupe
- 27 - Supprimer un groupe et tous ses attributs

3 - Gestion des équipement réseaux
- 31 - Ajouter un équipement réseau
- 32 - Supprimer un équipement réseau
- 33 - Lister un ou tous les équipements réseaux

4 - Administration
- 41 - Synchroniser les BD du serveur principal vers le secours
- 42 - Lancer une sauvegarde
- 43 - Afficher les tentatives de connexion erronées
- 44 - Afficher les tentatives de connexion réussies
- 45 - Afficher les dernières connexions au système
- 46 - Afficher l'état des serveurs
- 47 – Watchdog (services lancés ?)

La console d'administration peut être utilisée en mode intéractif, en mode automatique ou encore dans un mode mixte (on donne des arguments à la commande et les arguments manquants sont ensuite demandés de manière intéractive). Pour connaître les arguments possibles, taper :
`./netadmin.pl –help`

Synchronisation des BD

Attention ! Avec MySQL, le comportement diffèrera que vous mettiez *localhost* ou *127.0.0.1* ! Ne pas enlever ou ne pas intervertir les options indiquées dans le script !

Le code est accessible dans le code source de la console d'administration, dans la partie synchronisation des BD du serveur principal vers le serveur secondaire et dans la partie sauvegarde.
La sauvegarde est un dump dans un fichier (format *FichierSauvegarde_RADIUSAAAA_MM_JJ_hh_mm_ss*).

La synchronisation est un dump de la BD du serveur principal qui est desuite injecté dans la BD du serveur secondaire à travers un tunnel SSH afin que les informations ne transitent pas en clair sur le réseau.
Pour réinjecter une sauvegarde taper la commande suivante où *root* est l'utilisateur choisi pour administrer le SGBD et *FichierSauvegarde_RADIUS2007_10_3_16_21_40* le fichier de sauvegarde :
mysql radius –u root -p < FichierSauvegarde_RADIUS2007_10_3_16_21_40

Automatisation de la maintenance (sur le serveur principal)

Afin d'automatiser certaines tâches, on va utiliser le *cron* qui lancera *netadmin.pl* en mode non-intéractif ; c'est le mode **A**utomatique que l'on indique en précisant « *–A* »
au lancement du programme et en mettant en argument les valeurs qui seraient demandées en mode intéractif.
Pour configurer le *crond*, éditer le fichier */etc/crontab* pour y modifier les valeurs comme ci-dessous :

```
# /etc/crontab: system-wide crontab b you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
0 *    * * *   adminradius    cd / && run-parts --report /etc/cron.hourly
0 1    * * *   adminradius    test -x /usr/sbin/anacron || ( cd / && run-parts –report \
/etc/cron.daily )
0 2    * * 7   adminradius    test -x /usr/sbin/anacron || ( cd / && run-parts –report \
/etc/cron.weekly )
0 3    1 * *   adminradius    test -x /usr/sbin/anacron || ( cd / && run-parts –report \
/etc/cron.monthly )
#
```

Pour lancer toutes les semaines les tâches voulues, créer un fichier, avec les bons droits, dans */etc/cron.weekly* nommé *maintfreeradius* :

```
touch /etc/cron.weekly/maintfreeradius
chmod 755 /etc/cron.weekly/maintfreeradius
chown root.root /etc/cron.weekly/maintfreeradius
```

et y insérer le texte suivant en remplaçant *radiusXXX* et *radiusXXXMdP* par le login et le mot de passe de l'administrateur de la base de données :

```
#!/bin/sh
/adminradius/netadmin.pl –A –choix adm –sschoix sauv -LoginDB radiusXXX \
-PasswordDB  radiusXXXMdP
```

Pour lancer toutes les heures les tâches voulues, créer un fichier, avec les bons droits, dans */etc/cron.hourly* nommé *sync* :

```
touch /etc/cron.hourly/sync
chmod 755 /etc/cron.hourly/sync
chown root.root /etc/cron.hourly/sync
```

et y insérer le texte suivant :

```
# !/bin/sh
/adminradius/netadmin.pl –A -choix adm –sschoix sync
```

Pour lancer toutes les minutes la tâche voulue, rajouter dans le fichier */etc/crontab* la ligne suivante (si cela n'a pas déjà été fait lors de la configuration du watchdog) :

```
* * * * * adminradius sleep 30 && /adminradius/netadmin.pl –A –choix adm –sschoix watch
```

Nous allons recharger quotidiennement la configuration du firewall Netfilter pour palier la perte de prise de main à distance suite à une mauvaise manipulation des règles, ou pour reconfigurer le filtrage que l'on aurait oublié ouvert suite à des gestes de maintenance :

```
touch /etc/cron.daily/Firewall
chmod 755 /etc/cron.daily/Firewall
chown root.root /etc/cron.daily/Firewall
```

et insérer le texte suivant dans le fichier */etc/cron.daily/Firewall* :

```
# !/bin/sh
/etc/init.d/XXXXFirewall
```

optionnel : relancer *crond* avec la commande suivante :

```
/etc/init.d/cron restart
```

NOTE

La syntaxe du fichier */etc/crontab* est toujours la même.
Chaque entrée de la table (chaque ligne) correspond à une tâche à exécuter et est notée de la façon suivante:

mm hh jj MMM JJJ utilisateur tâche > log

Dans cette syntaxe :
mm représente la minute (de 0 à 59)
hh représente l'heure (de 0 à 23)
jj représente le numéro du jour du mois (de 1 à 31)
MMM représente le numéro du mois (de 1 à 12) ou l'abréviation du nom du mois (jan, feb, mar, apr, ...)
JJJ représente l'abréviation du nom du jour ou le chiffre correspondant au jour de la semaine.
(0 représente le dimanche, 1 représente le lundi, ...)
utilisateur utilisateur sous lequel est lancée la commande
tâche représente la commande ou le script shell à exécuter
log représente le nom d'un fichier dans lequel stocker le journal des opérations. On peut écraser (>) ou ajouter (>>). Si la clause > *log* n'est pas spécifiée, *cron* enverra automatiquement un mail de confirmation. Pour éviter cela il suffit de spécifier > */dev/null*

Par défaut, *crond* lance toutes les heures la commande « *cd / && run-parts --report /etc/cron.hourly* » qui a pour conséquence d'exécuter tous les scripts se trouvant dans */etc/cron.hourly*. Il en va de même pour tous les scripts se trouvant dans */etc/cron/daily* qui s'exécutent tous les jours, etc.

Automatisation de la maintenance (sur le serveur secondaire)

Lors de la synchronisation du serveur principal sur le serveur secondaire, les données de la base de données du premier écrasent celles de la base de données du second. Pour que certaines d'entres-elles soient prises en compte, il faut redémarrer FreeRADIUS. De plus, nous devons mettre en place le watchdog. D'ailleurs le temps de redémarrage de FreeRADIUS prend quelques secondes pendant lesquelles le watchdog se déclenche et détecte alors un incident (donc redémarrage de la machine la minute suivante). Pour éviter cela, par rapport au serveur principal, on rajoute *sleep 10 &&* sur la ligne concernant le watchdog, c'est-à-dire que nous attendons 10 secondes avant de vérifier les services. Pour tout cela, éditer le fichier */etc/crontab* pour y modifier les valeurs comme ci-dessous :

```
# /etc/crontab: system-wide crontab b you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
1 *     * * *   adminradius    cd / && run-parts --report /etc/cron.hourly
0 1     * * *   adminradius     test -x /usr/sbin/anacron || ( cd / && run-parts --report \ /
etc/cron.daily )
0 2     * * 7   adminradius     test -x /usr/sbin/anacron || ( cd / && run-parts --report \ /
etc/cron.weekly )
0 3     1 * *   adminradius     test -x /usr/sbin/anacron || ( cd / && run-parts --report \ /
etc/cron.monthly )
* *     * * *   adminradius  sleep 30 && /adminradius/netadmin.pl -A -choix adm -sschoix \
watch
1 *     * * *   root         /etc/init.d/freeradius restart
```

Comme sur le serveur principal, nous allons recharger quotidiennement la configuration du firewall Netfilter pour palier la perte de prise de main à distance suite à une mauvaise manipulation des règles, ou pour reconfigurer le filtrage que l'on aurait oublié ouvert suite à des gestes de maintenance :

```
touch /etc/cron.daily/Firewall
chmod 755 /etc/cron.daily/Firewall
chown root.root /etc/cron.daily/Firewall
```

et insérer le texte suivant dans le fichier */etc/cron.daily/Firewall* :

```
#!/bin/sh
/etc/init.d/XXXXFirewall
```

optionnel : relancer *crond* avec la commande suivante :
`/etc/init.d/cron restart`

Annexe : Script de création des tables dans la BD radius

```
##############################################################################
#  Copier les lignes dans un fichier que l'on nommera db_mysql.sql        #
#                                                                            #
#     Création de la base radius et de ses tables                            #
#                                                                            #
#     Pour charger le script :                                               #
#         mysql –u root < db_mysql.sql                                       #
#                                                                            #
##############################################################################

#
# Database creation and selection
#

CREATE DATABASE radius ;
USE radius ;

#
# Table structure for table 'radacct'
#

CREATE TABLE radacct (
  RadAcctId bigint(21) NOT NULL auto_increment,
  AcctSessionId varchar(32) NOT NULL default '',
  AcctUniqueId varchar(32) NOT NULL default '',
  UserName varchar(64) NOT NULL default '',
  Realm varchar(64) default '',
  NASIPAddress varchar(15) NOT NULL default '',
  NASPortId varchar(15) default NULL,
  NASPortType varchar(32) default NULL,
  AcctStartTime datetime NOT NULL default '0000-00-00 00:00:00',
  AcctStopTime datetime NOT NULL default '0000-00-00 00:00:00',
  AcctSessionTime int(12) default NULL,
  AcctAuthentic varchar(32) default NULL,
  ConnectInfo_start varchar(50) default NULL,
  ConnectInfo_stop varchar(50) default NULL,
  AcctInputOctets bigint(12) default NULL,
  AcctOutputOctets bigint(12) default NULL,
  CalledStationId varchar(50) NOT NULL default '',
  CallingStationId varchar(50) NOT NULL default '',
  AcctTerminateCause varchar(32) NOT NULL default '',
  ServiceType varchar(32) default NULL,
  FramedProtocol varchar(32) default NULL,
  FramedIPAddress varchar(15) NOT NULL default '',
  AcctStartDelay int(12) default NULL,
  AcctStopDelay int(12) default NULL,
  PRIMARY KEY  (RadAcctId),
  KEY UserName (UserName),
  KEY FramedIPAddress (FramedIPAddress),
  KEY AcctSessionId (AcctSessionId),
  KEY AcctUniqueId (AcctUniqueId),
  KEY AcctStartTime (AcctStartTime),
  KEY AcctStopTime (AcctStopTime),
  KEY NASIPAddress (NASIPAddress)
) ;

#
# Table structure for table 'radcheck'
#

CREATE TABLE radcheck (
  id int(11) unsigned NOT NULL auto_increment,
  UserName varchar(64) NOT NULL default '',
  Attribute varchar(32)  NOT NULL default '',
  op char(2) NOT NULL DEFAULT '==',
  Value varchar(253) NOT NULL default '',
  PRIMARY KEY  (id),
  KEY UserName (UserName(32))
) ;

#
# Table structure for table 'radgroupcheck'
#

CREATE TABLE radgroupcheck (
  id int(11) unsigned NOT NULL auto_increment,
  GroupName varchar(64) NOT NULL default '',
  Attribute varchar(32)  NOT NULL default '',
  op char(2) NOT NULL DEFAULT '==',
  Value varchar(253)  NOT NULL default '',
  PRIMARY KEY  (id),
  KEY GroupName (GroupName(32))
) ;

#
# Table structure for table 'radgroupreply'
#

CREATE TABLE radgroupreply (
  id int(11) unsigned NOT NULL auto_increment,
  GroupName varchar(64) NOT NULL default '',
  Attribute varchar(32)  NOT NULL default '',
  op char(2) NOT NULL DEFAULT '=',
  Value varchar(253)  NOT NULL default '',
  PRIMARY KEY  (id),
```

```
  KEY GroupName (GroupName(32))
) ;

#
# Table structure for table 'radreply'
#

CREATE TABLE radreply (
  id int(11) unsigned NOT NULL auto_increment,
  UserName varchar(64) NOT NULL default '',
  Attribute varchar(32) NOT NULL default '',
  op char(2) NOT NULL DEFAULT '=',
  Value varchar(253) NOT NULL default '',
  PRIMARY KEY  (id),
  KEY UserName (UserName(32))
) ;


#
# Table structure for table 'usergroup'
#

CREATE TABLE usergroup (
  UserName varchar(64) NOT NULL default '',
  GroupName varchar(64) NOT NULL default '',
  priority int(11) NOT NULL default '1',
  KEY UserName (UserName(32))
) ;

#
# Table structure for table 'radpostauth'
#

CREATE TABLE radpostauth (
  id int(11) NOT NULL auto_increment,
  user varchar(64) NOT NULL default '',
  pass varchar(64) NOT NULL default '',
  reply varchar(32) NOT NULL default '',
  date timestamp(14) NOT NULL,
  PRIMARY KEY  (id)
) ;

##########################################################################
#
#  The next table is commented out because it is not
#  currently used in the server.
#

#
# Table structure for table 'dictionary'
#
#CREATE TABLE dictionary (
#  id int(10) DEFAULT '0' NOT NULL auto_increment,
#  Type varchar(30),
#  Attribute varchar(64),
#  Value varchar(64),
#  Format varchar(20),
#  Vendor varchar(32),
#  PRIMARY KEY (id)
#);

#
# Table structure for table 'nas'
#
CREATE TABLE nas (
  id int(10) NOT NULL auto_increment,
  nasname varchar(128) NOT NULL,
  shortname varchar(32),
  type varchar(30) DEFAULT 'other',
  ports int(5),
  secret varchar(60) DEFAULT 'secret' NOT NULL,
  community varchar(50),
  groupe varchar(64), # Rajout pour supporter la fonctionnalité huntgroups
  description varchar(200) DEFAULT 'RADIUS Client',
  PRIMARY KEY (id),
  KEY nasname (nasname)
);

#
# Password and access level configuration
#

DELETE FROM mysql.user WHERE User='' ;
DELETE FROM mysql.user WHERE Host!='localhost' ;
DELETE FROM mysql.user WHERE User!='radiusXXX' AND User!='root' AND User!='debian-sys-maint' ;
GRANT ALL PRIVILEGES on *.* to root@localhost IDENTIFIED BY "MotDePasseRoot" ;
GRANT INSERT on radius.* to radiusXXX@localhost IDENTIFIED BY "radiusXXXMdP" ;
GRANT SELECT on radius.* to radiusXXX@localhost IDENTIFIED BY "radiusXXXMdP" ;
GRANT DELETE on radius.* to radiusXXX@localhost IDENTIFIED BY "radiusXXXMdP " ;
GRANT UPDATE on radius.* to radiusXXX@localhost IDENTIFIED BY "radiusXXXMdP " ;
# Les lignes « GRANT » ci-dessous ne sont utiles que dans le serveurs de secours.
GRANT DROP on radius.* to radiusXXX@localhost IDENTIFIED BY "radiusXXXMdP " ;
GRANT CREATE on radius.* to radiusXXX@localhost IDENTIFIED BY "radiusXXXMdP " ;
GRANT LOCK TABLES on radius.* to radiusXXX@localhost IDENTIFIED BY "radiusXXXMdP" ;
GRANT ALTER on radius.* to radiusXXX@localhost IDENTIFIED BY "radiusXXXMdP" ;
FLUSH PRIVILEGES ;
```

Annexe : Script de configuration du firewall sur serveur principal

```
#!/bin/sh

#modprobe fuse

echo "Chargement des modules" ;
# echo "Chargement du module fuse" ;
# modprobe fuse

echo "==> Chargement du module softdog" ;
modprobe softdog

echo "Chargement des modules terminé" ;

echo "Ouverture des flux" ;
iptables -F
iptables -Z
iptables -X

iptables -t mangle -F
iptables -t mangle -Z
iptables -t mangle -X

iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

echo "Mise à l'heure" ;
#ntpdate 172.16.64.1
hwclock --systohc # on met à l'heure l'horloge matérielle sur l'horloge système
echo "Mise à l'heure terminée"


echo "Fermeture des flux et définition des règles de filtrage" ;

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
echo "==> Application de la politique par défaut" ;

# Toute nouvelle connexion doit avoir un flag SYN.
iptables -A OUTPUT -p tcp ! --syn -m state --state NEW -j DROP

# Suppression des paquets invalides
iptables -A OUTPUT -p tcp -m state --state INVALID -j DROP

# Suppression des options IPv4 (reroutage, enregistrement, …)
#iptables -t mangle -A PREROUTING -j IPV4OPTSSTRIP

# Suppression des paquets non-unicasts
#iptables -A INPUT --pkt-type ! unicast -j DROP

# Protection contre le scan de ports
#iptables -A INPUT -m psd -j DROP

# Camoufle le serveur lors de traceroute
iptables -t mangle -A OUTPUT -j TTL --ttl-inc 1

# Ne répond pas aux TTL trop petit
iptables -A INPUT -m ttl --ttl-lt 4 -j DROP

# Suppression des paquets "sales"
# iptables -A INPUT -m unclean -j DROP

echo "==> Securisation diverses" ;

# Être SERVEUR SSH
iptables -A INPUT -s ! 172.16.69.17 -d 172.16.69.17 -p tcp --sport 1024: --dport 22 -i eth1 -j ACCEPT
iptables -A OUTPUT -s 172.16.69.17 -d ! 172.16.69.17 -p tcp --sport 22 --dport 1024: -o eth1 -m state --state ESTABLISHED,RELATED -j
ACCEPT
echo "==> Accès ouvert pour le serveur ssh" ;

# Être CLIENT SSH
iptables -A OUTPUT -s 172.16.69.17 -d ! 172.16.69.17 -p tcp --sport 1024: --dport 22 -o eth1 -j ACCEPT
iptables -A INPUT -s ! 172.16.69.17 -d 172.16.69.17 -p tcp --sport 22 --dport 1024: -i eth1 -m state --state ESTABLISHED,RELATED -j
ACCEPT
echo "==> Accès ouvert pour le client ssh" ;

# Être SERVEUR TELNET
#iptables -A OUTPUT -p tcp -s 172.16.69.17 -d ! 172.16.69.17 --sport 23 --dport 1024: -o eth1 -m state --state ESTABLISHED,RELATED -
j ACCEPT
#iptables -A INPUT -p tcp -s ! 172.16.69.17 -d 172.16.69.17 --sport 1024: --dport 23 -i eth1 -j ACCEPT
#echo "==> Accès ouvert pour le serveur telnet" ;

# OK pour du local
iptables -A INPUT -s 127.0.0.1 -d 127.0.0.1 -i lo -j ACCEPT
iptables -A OUTPUT -s 127.0.0.1 -d 127.0.0.1 -o lo -j ACCEPT
#iptables -A INPUT -s 172.16.69.17 -d 172.16.69.17 -i lo -j ACCEPT
#iptables -A OUTPUT -s 172.16.69.17 -d 172.16.69.17 -o lo -j ACCEPT
echo "==> Accès ouvert pour les traitements en local" ;

# Être CLIENT dns
#iptables -A OUTPUT -o eth1 -p udp --sport 1024: --dport 53 -j ACCEPT
#iptables -A INPUT -i eth1 -p udp --dport 1024: --sport 53 -m state --state ESTABLISHED,RELATED -j ACCEPT
#echo "==> Accès ouvert pour le client DNS" ;

# Être SERVEUR syslog
iptables -A INPUT -s ! 172.16.69.17 -d 172.16.69.17 -i eth1 -p udp --sport 514 --dport 514 -j ACCEPT
```

```
iptables -A OUTPUT -s 172.16.69.17 -d ! 172.16.69.17 -o eth1 -p udp --sport 514 --dport 514  -j ACCEPT
echo "==> Accès ouvert pour le serveur syslog" ;

# Être SERVEUR ntp
iptables -A INPUT -s ! 172.16.69.17 -d 172.16.69.17 -i eth1 -p udp --dport 123 -j ACCEPT
iptables -A OUTPUT -s 172.16.69.17 -d ! 172.16.69.17 -o eth1 -p udp --sport 123 -m state --state ESTABLISHED,RELATED -j ACCEPT
echo "==> Accès ouvert pour le serveur ntp" ;

# Être CLIENT ntp
iptables -A INPUT -s ! 172.16.69.17 -d 172.16.69.17 -i eth1 -p udp --sport 123: --dport 1024: -m state --state ESTABLISHED,RELATED -
j ACCEPT
iptables -A OUTPUT -s 172.16.69.17 -d ! 172.16.69.17 -o eth1 -p udp --sport 1024: --dport 123 -j ACCEPT
echo "==> Accès ouvert pour le client ntp" ;

# Être SERVEUR radius
iptables -A INPUT -i eth1 -s ! 172.16.69.17 -d 172.16.69.17 -p udp --sport 1024: --dport 1812 -j ACCEPT
iptables -A OUTPUT -o eth1 -s 172.16.69.17 -d ! 172.16.69.17 -p udp --sport 1812 --dport 1024: -j ACCEPT
iptables -A INPUT -i eth1 -s ! 172.16.69.17 -d 172.16.69.17 -p udp --sport 1024: --dport 1813 -j ACCEPT
iptables -A OUTPUT -o eth1 -s 172.16.69.17 -d ! 172.16.69.17 -p udp --sport 1813 --dport 1024: -j ACCEPT
echo "==> Accès ouvert pour le serveur radius" ;

# icmp
iptables -A INPUT -p icmp --icmp-type ping -m length --length 1000 -j DROP

iptables -A INPUT -p icmp --icmp-type pong -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type network-unreachable -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type host-unreachable -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type protocol-unreachable -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type fragmentation-needed -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type ping -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type ttl-exceeded -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 30 -i eth1 -j ACCEPT # traceroute

iptables -A OUTPUT -p icmp --icmp-type pong -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type network-unreachable -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type host-unreachable -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type protocol-unreachable -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type fragmentation-needed -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type ping -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type ttl-exceeded -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 30 -o eth1 -j ACCEPT # traceroute

iptables -A INPUT -p icmp -j DROP
iptables -A OUTPUT -p icmp -j DROP
echo "==> Accès ouvert pour divers paquet icmp" ;

route add -net 172.16.0.0 netmask 255.255.240.0 gw 172.16.69.3
route add -net 172.16.16.0 netmask 255.255.240.0 gw 172.16.69.3
echo "==> Ajout des règles de routage" ;

echo "Configuration du firewall terminée" ;

echo "Affichage des journaux système et FreeRADIUS en consoles 8 et 9" ;
killall lwatch
killall cat
killall tail
cat /dev/xconsole | lwatch -i - >> /dev/tty8 &
tail -f /var/log/freeradius/radius.log | lwatch -i - >> /dev/tty9 &
echo "Affichage des journaux système réalisé" ;
```

Annexe : Script de configuration du firewall sur serveur de secours

```
#!/bin/sh

#modprobe fuse

echo "Chargement des modules" ;
# echo "Chargement du module fuse" ;
# modprobe fuse

echo "==> Chargement du module softdog" ;
modprobe softdog

echo "Chargement des modules terminé" ;

echo "Ouverture des flux" ;
iptables -F
iptables -Z
iptables -X

iptables -t mangle -F
iptables -t mangle -Z
iptables -t mangle -X

iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

echo "Mise à l'heure" ;
#ntpdate 172.16.64.1
hwclock --systohc # on met à l'heure l'horloge matérielle sur l'horloge système
echo "Mise à l'heure terminée"


echo "Fermeture des flux et définition des règles de filtrage" ;

iptables -P INPUT DROP
iptables -P OUTPUT DROP
```

```
iptables -P FORWARD DROP
echo "==> Application de la politique par défaut" ;

# Toute nouvelle connexion doit avoir un flag SYN.
iptables -A OUTPUT -p tcp ! --syn -m state --state NEW -j DROP

# Suppression des paquets invalides
iptables -A OUTPUT -p tcp -m state --state INVALID -j DROP

# Suppression des options IPv4 (reroutage, enregistrement, …)
#iptables -t mangle -A PREROUTING -j IPV4OPTSSTRIP

# Suppression des paquets non-unicasts
#iptables -A INPUT --pkt-type ! unicast -j DROP

# Protection contre le scan de ports
#iptables -A INPUT -m psd -j DROP

# Camoufle le serveur lors de traceroute
iptables -t mangle -A OUTPUT -j TTL --ttl-inc 1

# Ne répond pas aux TTL trop petit
iptables -A INPUT -m ttl --ttl-lt 4 -j DROP

# Suppression des paquets "sales"
# iptables -A INPUT -m unclean -j DROP

echo "==> Securisation diverses" ;

# Être SERVEUR SSH
iptables -A INPUT -s ! 172.16.80.230 -d 172.16.80.230 -p tcp --sport 1024: --dport 22 -i eth1 -j ACCEPT
iptables -A OUTPUT -s 172.16.80.230 -d ! 172.16.80.230 -p tcp --sport 22 --dport 1024: -o eth1 -m state --state ESTABLISHED,RELATED
-j ACCEPT
echo "==> Accès ouvert pour le serveur ssh" ;

# Être CLIENT SSH
iptables -A OUTPUT -s 172.16.80.230 -d ! 172.16.80.230 -p tcp --sport 1024: --dport 22 -o eth1 -j ACCEPT
iptables -A INPUT -s ! 172.16.80.230 -d 172.16.80.230 -p tcp --sport 22 --dport 1024: -i eth1 -m state --state ESTABLISHED,RELATED -
j ACCEPT
echo "==> Accès ouvert pour le client ssh" ;

# Être SERVEUR TELNET
#iptables -A OUTPUT -p tcp -s 172.16.80.230 -d ! 172.16.80.230 --sport 23 --dport 1024: -o eth1 -m state --state ESTABLISHED,RELATED
-j ACCEPT
#iptables -A INPUT -p tcp -s ! 172.16.80.230 -d 172.16.80.230 --sport 1024: --dport 23 -i eth1 -j ACCEPT
#echo "==> Accès ouvert pour le serveur telnet" ;

# OK pour du local
iptables -A INPUT -s 127.0.0.1 -d 127.0.0.1 -i lo -j ACCEPT
iptables -A OUTPUT -s 127.0.0.1 -d 127.0.0.1 -o lo -j ACCEPT
#iptables -A INPUT -s 172.16.80.230 -d 172.16.80.230 -i lo -j ACCEPT
#iptables -A OUTPUT -s 172.16.80.230 -d 172.16.80.230 -o lo -j ACCEPT
echo "==> Accès ouvert pour les traitements en local" ;

# Être CLIENT dns
#iptables -A OUTPUT -o eth1 -p udp --sport 1024: --dport 53 -j ACCEPT
#iptables -A INPUT -i eth1 -p udp --dport 1024: --sport 53 -m state --state ESTABLISHED,RELATED -j ACCEPT
#echo "==> Accès ouvert pour le client DNS" ;

# Être CLIENT ntp
iptables -A INPUT -s ! 172.16.80.230 -d 172.16.80.230 -i eth1 -p udp --sport 123: --dport 1024: -m state --state ESTABLISHED,RELATED
-j ACCEPT
iptables -A OUTPUT -s 172.16.80.230 -d ! 172.16.80.230 -o eth1 -p udp --sport 1024: --dport 123 -j ACCEPT
echo "==> Accès ouvert pour le client ntp" ;

# Être SERVEUR radius
iptables -A INPUT -i eth1 -s ! 172.16.80.230 -d 172.16.80.230 -p udp --sport 1024: --dport 1812 -j ACCEPT
iptables -A OUTPUT -o eth1 -s 172.16.80.230 -d ! 172.16.80.230 -p udp --sport 1812 --dport 1024: -j ACCEPT
iptables -A INPUT -i eth1 -s ! 172.16.80.230 -d 172.16.80.230 -p udp --sport 1024: --dport 1813 -j ACCEPT
iptables -A OUTPUT -o eth1 -s 172.16.80.230 -d ! 172.16.80.230 -p udp --sport 1813 --dport 1024: -j ACCEPT
echo "==> Accès ouvert pour le serveur radius" ;

# icmp
iptables -A INPUT -p icmp --icmp-type ping -m length --length 1000 -j DROP

iptables -A INPUT -p icmp --icmp-type pong -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type network-unreachable -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type host-unreachable -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type protocol-unreachable -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type fragmentation-needed -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type ping -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type ttl-exceeded -i eth1 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 30 -i eth1 -j ACCEPT # traceroute

iptables -A OUTPUT -p icmp --icmp-type pong -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type network-unreachable -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type host-unreachable -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type protocol-unreachable -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type fragmentation-needed -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type ping -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type ttl-exceeded -o eth1 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 30 -o eth1 -j ACCEPT # traceroute

iptables -A INPUT -p icmp -j DROP
iptables -A OUTPUT -p icmp -j DROP
echo "==> Accès ouvert pour divers paquet icmp" ;

route add -net 172.16.0.0 netmask 255.255.240.0 gw 172.16.69.3
route add -net 172.16.16.0 netmask 255.255.240.0 gw 172.16.69.3
echo "==> Ajout des règles de routage" ;
```

```
echo "Configuration du firewall terminée" ;
```

Annexe : Code source de la console d’administration

```
#!/usr/bin/perl

# v1.0 Programme de Guillaume Lehmann. Programme protégé par la licence GNU Public Licence version 3.

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# CONSOLE D'ADMINISTRATION POUR FREERADIUS #
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#


use DBI ;
use Time::localtime ;
use Getopt::Long ;



# Définition de tous les formatages
format ListUsers =
|@|||||||||||| |@|||||||||||||||||| |@|||||||||||||||| |
$ResReqLign->{'UserName'}, $ResReqLign->{'Value'}, $ResReqLign->{'GroupName'}
.

    format ListReqRep =
|@|||| |@||||||||||| |@|||||||||||||||||||||||| |@|||||| |@|||||||||||||||||||||||| |
$ResReqLign->{'id'}, $ResReqLign->{'GroupName'}, $ResReqLign->{'Attribute'}, $ResReqLign->{'op'}, $ResReqLign->{'Value'}
.
    format ListNas =
|@||||||||||||| |@||||||||||||| |@|||||| |@||| |@||||||||||||| |@|||||||||| |@||||||| |@|||||||||||||||||| |
$ResReqLign->{'nasname'}, $ResReqLign->{'shortname'}, $ResReqLign->{'type'}, $ResReqLign->{'ports'}, $ResReqLign->{'secret'}, $ResReqLign->{'groupe'},
$ResReqLign->{'community'}, $ResReqLign->{'description'}
.

    format ListConnOK =
|@|||| |@|||||||||||||| |@||||||||||||||||||| |@||||||||||||||||||| |
$ResReqLign->{'id'}, $ResReqLign->{'user'}, $ResReqLign->{'reply'}, $ResReqLign->{'date'}
.


# Récupération des éventuels arguments de la commande.
    GetOptions("A!" => \$Option, "choix=s" => \$Choix, "sschoix=s" => \$SousChoix, "nom:s" => \$Nom, "mdp:s" => \$MdP, "groupe:s" => \$Group, "newmdp:s" =>
\$NewMdP, "attribute:s" => \$Attribute, "op:s" => \$Op, "value:s" => \$Value, "nasname:s" => \$Nasname, "shortname:s" => \$Shortname, "type:s" => \$Type,
"port:i" => \$Ports, "secret:s" => \$Secret, "description:s" => \$Description, "help" => \$Help) ;

# Affichage du menu d'aide si --help est indiqué.
if ($Help) {
    print "\n                                          -==Introduction==-" ;
    print "\nL'ordre des options importe peu. Il n'est pas nécessaire de mettre en entier le nom des options (juste assez de lettres que ce ne soit pas
ambigüe). En revanche les valeurs des options ne doivent pas être abrégées.\n" ;
    print "Mettre \"-\" ou \"--\" devant les noms des options revient au même.\n" ;
    print "La casse du nom des options n'a pas d'influence.\n\n" ;
    print "                                            -==Arguments==-\n" ;
    print "--choix [1|user, 2|group, 3|nas, 4|adm] : choix dans menu.\n" ;
    print "--sschoix [11|adduser, 12|del, 13|modpass, 14|addatt, 15|list]\n" ;
    print "          [21|addreq, 22|addrep, 23|listreq, 24|listrep, 25|delreq, 26|delrep,  27|delgroup]\n" ;
    print "          [31|add, 32|del, 33|list]\n" ;
    print "          [41|sync, 42|sauv, 43|listerr, 44|listok, 45|last, 46|state, 47|watch]\n" ;
    print "    choix dans sous-menu.\n" ;
    print "--nom : nom\n" ;
    print "--mdp : mot de passe\n" ;
    print "--groupe : groupe\n" ;
    print "--newmdp : nouveau mot de passe lors d'un changement de mot de passe (l'actuel est indiqué alors par --MdP).\n" ;
    print "--attribute : nom de l'attribut manipulé\n" ;
    print "--op : opérateur qui sera appliqué à l'attribut --Attribute\n" ;
    print "--value : valeur d'un attribut --Attribute en fonction de l'opérateur --Op.\n" ;
    print "--nasname : élément identifiant l'équipement réseau (généralement son adresse IP).\n" ;
    print "--shortname : nom court de l'équipement réseau.\n" ;
    print "--type : type d'équipement (\"other\" pour un switch).\n" ;
    print "--port : port de l'équipement réseau duquel doit venir la requête (0 par défaut).\n" ;
    print "--secret : secret partagé à la création d'un équipement réseau.\n" ;
    print "--description : description ou commentaire à la création d'un équipement réseau.\n" ;
    print "--loginrootdb : login de l'administrateur de la base de données.\n" ;
    print "--passwordrootdb : mot de passe de l'administrateur de la base de données.\n" ;
    print "--help : affiche le menu présent.\n\n" ;
    print "                             -==Listing des arguments pour chaque menu==-\n" ;
    print "Menu 11 : --nom  --mdp  --groupe\n" ;
    print "Menu 12 : --nom  --mdp\n" ;
    print "Menu 13 : --nom  --newmdp\n" ;
    print "Menu 14 : --nom  --attribute  --op  --value  --mdp\n" ;
    print "Menu 15 : Pas d'argument si on veut tout lister, sinon --nom\n" ;
    print "Menu 21 : --groupe  --attribute  --op  --value\n" ;
    print "Menu 22 : --groupe  --attribute  --op  --value\n" ;
    print "Menu 23 : Pas d'argument\n" ;
    print "Menu 24 : Pas d'argument\n" ;
    print "Menu 25 : --groupe  --attribute\n" ;
    print "Menu 26 : --groupe  --attribute\n" ;
    print "Menu 27 : --groupe  --mdp\n" ;
    print "Menu 31 : --nasname  --shortname  --type  --port  --secret  --groupe  --description\n" ;
    print "Menu 32 : --nasname  --mdp\n" ;
    print "Menu 33 : Pas d'argument si on veut tout lister, sinon --nom\n" ;
    print "Menu 41 : Pas d'argument\n" ;
    print "Menu 42 : --logindb  --passworddb\n" ;
    print "Menu 43 : Pas d'argument\n" ;
    print "Menu 44 : --mdp\n" ;
    print "Menu 45 : Pas d'argument\n" ;
    print "Menu 46 : Pas d'argument\n" ;
    print "Menu 47 : Pas d'argument\n" ;
    print "\nProgramme sous licence GPLv3. Guillaume Lehmann\n" ;
    exit(1) ;
}

# Si -A n'est pas rentré comme option, alors on n'est pas en mode Automatique mais en mode Intéractif.
if (!$Option) {
    # Initialisation de la variable pour pouvoir rentrer dans la boucle while.
    $Choix = "1" ;
}
else {
    $reponse = "o" ;
}
```

```
while ($Choix ne "0") {
    if (!$Option) {
          $reponse = "" ;
    }
    else {
          $reponse = "o" ;
    }


    # Affichage du menu si on n'est pas en mode Automatique.
    if (!$Option) {
          print "\n\n##############################################\n" ;
          print "#### CONSOLE D'ADMINISTRATION FreeRADIUS ####\n" ;
              print " 0 - Quitter l'application\n" ;
          print " 1 - Gestion des utilisateurs\n" ;
          print " 2 - Gestion des groupes\n" ;
          print " 3 - Gestion des équipements réseaux\n" ;
          print " 4 - Administration\n" ;
          print "##############################################\n" ;
    }
    # Récupération des identifiants pour accéder à la base de données.
    open(ACCESBD, '/adminradius/AccesBD.txt') || die "Problème à l'accès au fichier AccesBD.txt" ;
#    open(ACCESBD, 'AccesBD.txt') || die "Problème à l'accès au fichier AccesBD.txt" ;
    $resultat = <ACCESBD> ;
    close (ACCESBD) ;
    ($DatabaseDB, $HostnameDB, $LoginDB, $PasswordDB, $TypDB) = $resultat =~ /^(\w*)\s*(\w*)\s*(\w*)\s*(\w*)\s*(\w*)\s*$/ ;

    $dsn = "DBI:$TypDB:database=$DatabaseDB;host=$HostnameDB" ;

    # Ouverture de la base de données.
    $dbh = DBI->connect($dsn, $LoginDB, $PasswordDB) ;

    # Lecture du choix de l'utilisateur en mode Intéractif
    if (!$Option) {
          print "Choix : " ;
          $Choix = <STDIN> ;
          chomp $Choix ;
    }


    # GESTION DES UTILISATEURS
    if ($Choix eq "1" or $Choix eq "user") {
          # Affichage du menu si l'on n'est pas en mode Automatique.
          if (!$Option) {
              print "\n\n##############################################\n" ;
              print "11 - Ajouter un utilisateur\n" ;
              print "12 - Supprimer un utilisateur\n" ;
              print "13 - Modifier le mot de passe d'un utilisateur\n" ;
              print "14 - Ajout d'un attribut à un utilisateur\n" ;
              print "15 - Lister un ou tous les utilisateurs\n" ;
              print "##############################################\n" ;

              # Lecture du choix de l'utilisateur en mode Intéractif
              print "Choix : " ;
              $SousChoix = <STDIN> ;
              chomp $SousChoix ;
          }
          if ($SousChoix eq "11" or $SousChoix eq "adduser") {
              # Récupérer les informations en mode Intéractif, sinon en mode Automatique utiliser les arguments de la commande
              if (!$Nom) {
                    print "Nom : " ;
                    $Nom = <STDIN> ;
                    chomp $Nom ;
              }
              if (!$MdP) {
                    print "\nMot de passe : " ;
                    $MdP = <STDIN> ;
                    chomp $MdP ;
              }
              if (!$Group) {
                    print "\nGroupe : " ;
                    $Group = <STDIN> ;
                    chomp $Group ;
              }

              # Rajouter un utilisateurs dans la base de données après avoir vérifié qu'il n'existe pas déjà et que sont login et mdp ne sont pas nuls
              if ($Nom eq "" or $MdP eq "") {
                    # Ne rien faire car le login ou le mot de passe est vide.
                    print "[NOK] Le login ou le mot de passe est vide\n" ;
              }
              else {
                    $ResReq = $dbh->prepare("SELECT UserName FROM radcheck WHERE UserName LIKE \"$Nom\";") ;
                    $ResReq->execute() ;
                    $ResReqLign = $ResReq->fetchrow_hashref() ;
                    if ($ResReqLign->{'UserName'} ne "") {
                        # Ne rien faire car l'utilisateur existe déjà
                        print "[NOK] L'utilisateur existe déjà\n" ;
                        $ResReq->finish() ;
                    }
                    else {
                        # Nettoyer requête.
                        $ResReq->finish() ;
                        # Création des paramètres de création du mot de passe système.
                        $chaine = './0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
                        # Création de la graine.
                        for ($i = 0; $i < 8; $i++) {
                              $graine .= substr($chaine, rand(length($chaine)), 1);
                        }
                        # Création du mot de passe chiffré au format MD5 (donc graine au format $1$ suivi de 8 caract$
                        $mdpCrypt = crypt($MdP, '$1$'.$graine.'$') ;

                        # Faire l'insertion.
                        $ResReq = $dbh->do("INSERT INTO radcheck (UserName, Attribute, op, Value) VALUES (\"$Nom\", \"Crypt-
Password\", \"==\", \"$MdP\") ;") ;
                        $ResReq = $dbh->do("INSERT INTO usergroup (UserName, GroupName) VALUES (\"$Nom\", \"$Group\") ;") ;
                        print "[OK] Utilisateur créé sous FreeRADIUS\n\n" ;
                        sleep 1 ;

                        # Création de l'utilisateur dans le système.
                        print `adduser --system --home /home/RADIUSuser --shell /adminradius/chroot --ingroup RADIUSgroup $Nom` ;
                        # Changement/Définition du mot de passe.
                        print `usermod -p '$mdpCrypt' $Nom` ;
                        print "Sauf si erreur ci-dessus, utilisateur créé dans le système\n" ;
                        sleep 1 ;
                    } # Fin if ($ResReqLign->{'UserName'} ne "") ELSE
              } # Fin if ($Nom eq "" or $MdP eq "") {} ELSE
          } # Fin if ($SousChoix eq "11")
```

```
if ($SousChoix eq "12" or $SousChoix eq "del") {
    # SUPPRIMER UN UTILISATEUR
    # Récupérer les informations en mode Intéractif, sinon en mode Automatique utiliser les arguments de la commande
    if (!$Nom) {
          print "Nom : " ;
          $Nom = <STDIN> ;
          chomp $Nom ;
    }
    if (!$MdP) {
          print "\nMot de passe de l'utilisateur de la base : " ;
          $MdP = <STDIN> ;
          chomp $MdP ;
    }
    if (!$reponse) {
          # Demander une confirmation avant de faire l'irrémédiable.
          print "Confirmer suppression ? (o/n)" ;
          $reponse = <STDIN> ;
          chomp $reponse ;
    }

    # Il faut le mot de passe user valide et la confirmation de la suppression pour l'effectuer.
    if (($reponse eq "o") and ($MdP eq $PasswordDB)) {
          # Supprimer l'utilisateur dans la base de données.
          $ResReq = $dbh->do("DELETE FROM radcheck WHERE UserName LIKE \"$Nom\" ;") ;
          $ResReq = $dbh->do("DELETE FROM radreply WHERE UserName LIKE \"$Nom\" ;") ;
          $ResReq = $dbh->do("DELETE FROM usergroup WHERE UserName LIKE \"$Nom\" ;") ;
          print "\n[OK] Suppression effectuée sous FreeRADIUS\n\n" ;

          print `userdel $Nom` ;
          print "[OK] Suppression effectuée dans le système\n" ;
          sleep 1 ;
    }
    else {
          # Ne rien faire.
          print "\n[NOK] Suppression non-réalisée (mauvais mot de passe ou action suspendue par l'utilisateur)\n" ;
    }

} # Fin if ($SousChoix eq "12")

if ($SousChoix eq "13" or $SousChoix eq "motpass") {
    # MODIFIER LE MOT DE PASSE D'UN UTILISATEUR
    # Récupérer les informations en mode Intéractif, sinon en mode Automatique utiliser les arguments de la commande
    if (!$Nom) {
          print "Nom : " ;
          $Nom = <STDIN> ;
          chomp $Nom ;
    }
    if (!$NewMdP) {
          print "\nNouveau mot de passe : " ;
          $NewMdP = <STDIN> ;
          chomp $NewMdP ;
    }

    # Création des paramètres de création du mot de passe système.
    $chaine = './0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
    # Création de la graine.
    for ($i = 0; $i < 8; $i++) {
          $graine .= substr($chaine, rand(length($chaine)), 1);
    }
    # Création du mot de passe chiffré au format MD5 (donc graine au format $1$ suivi de 8 caractères puis de $).
    $mdpCrypt = crypt($NewMdP, '$1$'.$graine.'$');

    # Si on trouve l'utilisateur avec son ancien mot de passe, alors on met à jour son mot de passe.
    $ResReq = $dbh->prepare("SELECT UserName FROM radcheck WHERE UserName LIKE \"$Nom\" ;") ;
    $ResReq->execute() ;

    $ResReqLign = $ResReq->fetchrow_hashref() ;
    if ($ResReqLign->{'UserName'} eq "") {
          # Ne rien faire car l'utilisateur n'existe pas ou le mot de passe n'est pas le bon.
          print "[NOK] Impossible de trouve l'utilisateur.\n" ;
          $ResReq->finish() ;
    }
    else {
          # Nettoyer requête et faire la mise à jour du mot de passe.
          $ResReq->finish() ;
          $ResReq = $dbh->do("UPDATE radcheck SET Value = \"$mdpCrypt\", Attribute = \"Crypt-Password\" WHERE UserName LIKE \"$Nom\" ;") ;

          print "[OK] Modification de mot de passe réalisée pour FreeRADIUS\n\n" ;

          # Changement/Définition du mot de passe.
          print `usermod -p '$mdpCrypt' $Nom` ;
          print "Sauf si erreur ci-dessus (et non seulement des warnings), le mot de passe est également modifié dans le système\n" ;
          sleep 1;
    }
} # Fin if ($SousChoix eq "13")

if ($SousChoix eq "14" or $SousChoix eq "addatt") {
    # AJOUTER UN ATTRIBUT A UN UTILISATEUR
    # Récupérer les informations en mode Intéractif, sinon en mode Automatique utiliser les arguments de la commande
    if (!$Nom) {
          print "Nom : " ;
          $Nom = <STDIN> ;
          chomp $Nom ;
    }
    if (!$Attribute) {
          print "\nAttribut (Password, ...) : " ;
          $Attribute = <STDIN> ;
          chomp $Attribute ;
    }
    if (!$Op) {
          print "\nOpérateur : " ;
          $Op = <STDIN> ;
          chomp $Op ;
    }
    if (!$Value) {
          print "\nValeur attribut : ";
          $Value = <STDIN> ;
          chomp $Value ;
    }
    if (!$MdP) {
          print "\nMot de passe de l'utilisateur de la base :" ;
          $MdP = <STDIN> ;
          chomp $MdP ;
    }
    # Il faut que le mot de passe du user soit correct pour effectuer l'insertion.
    if ($MdP eq $PasswordDB) {
          $ResReq = $dbh->do("INSERT INTO radcheck (UserName, Attribute, op, Value) VALUES (\"$Nom\", \"$Attribute\", \"$Op\", \"$Value\") ;") ;
```

```
                print "[OK] Insertion effectuée\n" ;
                sleep 1 ;
            }
            else {
                # Ne rien faire, le mot de passe user n'est pas le bon.
                print "[NOK] Action non-réalisée, le mot de passe utilisateur n'est pas le bon\n" ;
            }
        } # Fin if ($SousChoix eq "14")

        if ($SousChoix eq "15" or $SousChoix eq "list") {
            # LISTER LES UTILISATEURS
            # .. Demande de l'utilisateur à afficher si l'on n'est pas en mode Automatique
            if (!$Option) {
                print "Nom (laisser vide pour tout afficher) : " ;
                $Nom = <STDIN> ;
                chomp $Nom ;
            }
            # .. Si on a un login, on ne liste pas tout.
            if ($Nom =~ /\w+/) {
                $ResReq = $dbh->prepare("SELECT radcheck.UserName, radcheck.Attribute, radcheck.op, radcheck.Value, usergroup.GroupName FROM radcheck,
usergroup WHERE radcheck.UserName = usergroup.UserName AND radcheck.UserName LIKE \"$Nom\" ;") ;
            }
            else {
                $ResReq = $dbh->prepare("SELECT radcheck.UserName, radcheck.Attribute, radcheck.op, radcheck.Value, usergroup.GroupName FROM radcheck,
usergroup WHERE radcheck.UserName = usergroup.UserName ;") ;
            }
            $ResReq->execute() ;
            # Ligne de titre
            print "------------------------------------------------------------\n" ;
            print "|     Nom       |     Mot de passe    |      Groupe       |\n" ;
            print "------------------------------------------------------------\n" ;
            # Résultat
            $~ = "ListUsers" ;
            while ($ResReqLign = $ResReq->fetchrow_hashref()) {
                write() ;
            }
            print "------------------------------------------------------------\n" ;
            # Nettoyer la requête
            $ResReq->finish() ;
        } # Fin if ($SousChoix eq "15")

    } # Fin if (Choix eq "1")


    if ($Choix eq "2" or $Choix eq "group") {
        # Affichage du menu si l'on n'est pas en mode Automatique.
        if (!$Option) {
            print "\n\n##############################################\n" ;
            print "21 - Ajouter un attribut de requête à un groupe\n" ;
            print "22 - Ajouter un attribut de réponse à un groupe\n" ;
            print "23 - Lister les attributs de requête\n" ;
            print "24 - Lister les attributs de réponse\n" ;
            print "25 - Supprimer un attribut de requête à un groupe\n" ;
            print "26 - Supprimer un attribut de réponse à un groupe\n" ;
            print "27 - Supprimer un groupe et tous ses attributs\n" ;
            print "##############################################\n" ;

            # Lecture du choix de l'utilisateur en mode Intéractif
            print "Choix : " ;
            $SousChoix = <STDIN> ;
            chomp $SousChoix ;
        }

        if ($SousChoix eq "21" or $SousChoix eq "addreq") {
            # AJOUTER UN ATTRIBUT DE REQUÊTE À UN GROUPE
            # Récupérer les informations en mode Intéractif, sinon en mode Automatique utiliser les arguments de la commande
            if (!$Group) {
                print "Groupe : " ;
                $Group = <STDIN> ;
                chomp $Group ;
            }
            if (!$Attribute) {
                print "\nAttribut : " ;
                $Attribute = <STDIN> ;
                chomp $Attribute ;
            }
            if (!$Op) {
                print "Opérateur : " ;
                $Op = <STDIN> ;
                chomp $Op ;
            }
            if (!$Value) {
                print "Valeur de l'attribut : " ;
                $Value = <STDIN> ;
                chomp $Value ;
            }
            # Réaliser l'insertion.
            $ResReq = $dbh->do("INSERT INTO radgroupcheck (GroupName, Attribute, op, Value) VALUES
(\"$Group\", \"$Attribute\", \"$Op\", \"$Value\") ;") ;
            print "[OK] Insertion effectuée\n" ;
            sleep 1 ;

        } # Fin if ($SousChoix eq "21" or $SousChoix eq "addreq")

        if ($SousChoix eq "22" or $SousChoix eq "addrep") {
            # AJOUTER UN ATTRIBUT DE RÉPONSE À UN GROUPE
            # Récupérer les informations en mode Intéractif, sinon en mode Automatique utiliser les arguments de la commande
            if (!$Group) {
                print "Groupe : " ;
                $Group = <STDIN> ;
                chomp $Group ;
            }
            if (!$Attribute) {
                print "\nAttribut : " ;
                $Attribute = <STDIN> ;
                chomp $Attribute ;
            }
            if (!$Op) {
                print "Opérateur : " ;
                $Op = <STDIN> ;
                chomp $Op ;
            }
            if (!$Value) {
                print "Valeur de l'attribut : " ;
                $Value = <STDIN> ;
                chomp $Value ;
            }
            # Réaliser l'insertion.
```

```
    $ResReq = $dbh->do("INSERT INTO radgroupreply(GroupName, Attribute, op, Value) VALUES (\"$Group\", \"$Attribute\", \"$Op\", \"$Value\") ;") ;
    print "[OK] Insertion effectuée\n" ;
    sleep 1 ;
} # Fin du if ($SousChoix eq "22" or $SousChoix eq "addrep")

if ($SousChoix eq "23" or $SousChoix eq "listreq") {
    # LISTER LES ATTRIBUTS DE REQUÊTE
    # Construction et exécution de la requête.
    $ResReq = $dbh->prepare("SELECT * FROM radgroupcheck ;") ;
    $ResReq->execute() ;
    # Ligne de titre
    print "-------------------------------------------------------------------------------------------\n" ;
    print "|  id  |    Groupe    |          Attribut          |Opérateur|           Valeur          |\n" ;
    print "-------------------------------------------------------------------------------------------\n" ;
    # Résultat
    $~ = "ListReqRep" ;
    while ($ResReqLign = $ResReq->fetchrow_hashref()) {
          write() ;
    }
    print "-------------------------------------------------------------------------------------------\n" ;
    # Nettoyer la requête
    $ResReq->finish() ;
} # Fin if ($SousChoix eq "23" or $SousChoix eq "listreq")

if ($SousChoix eq "24" or $SousChoix eq "listrep") {
    # LISTER LES ATTRIBUTS DE RÉPONSE
    # Construction et exécution de la requête.
    $ResReq = $dbh->prepare("SELECT * FROM radgroupreply ;") ;
    $ResReq->execute() ;
    # Ligne de titre
    print "-------------------------------------------------------------------------------------------\n" ;
    print "|  id  |    Groupe    |          Attribut          |Opérateur|           Valeur          |\n" ;
    print "-------------------------------------------------------------------------------------------\n" ;
    # Résultat
    $~ = "ListReqRep" ;
    while ($ResReqLign = $ResReq->fetchrow_hashref()) {
          write() ;
    }
    print "-------------------------------------------------------------------------------------------\n" ;
    # Nettoyer la requête
    $ResReq->finish() ;
} # Fin if ($SousChoix eq "24" or $SousChoix eq "listrep")

if ($SousChoix eq "25" or $SousChoix eq "delreq") {
    # SUPPRIMER UN ATTRIBUT DE REQUÊTE À UN GROUPE
    # Récupérer les informations en mode Intéractif, sinon en mode Automatique utiliser les arguments de la commande
    if (!$Group) {
          print "Groupe : " ;
          $Group = <STDIN> ;
          chomp $Group ;
    }
    if (!$Attribute) {
          print "\nAttribut : " ;
          $Attribute = <STDIN> ;
          chomp $Attribute ;
    }
    if (!$reponse) {
          # Demander une confirmation avant de faire l'irrémédiable.
          print "Confirmer suppression ? (o/n)" ;
          $reponse = <STDIN> ;
          chomp $reponse ;
    }
    # Réaliser la suppression.
    if ($reponse eq "o") {
          $ResReq = $dbh->do("DELETE FROM radgroupcheck WHERE GroupName=\"$Group\" AND Attribute=\"$Attribute\" ;") ;
          print "[OK] Suppression effectuée\n" ;
          sleep 1 ;
    }
    else {
          # Ne rien faire
          print "\n[NOK] Suppression non-réalisée (action suspendue par l'utilisateur)\n" ;
    }
} # Fin du if ($SousChoix eq "25" or $SousChoix eq "delreq")

if ($SousChoix eq "26" or $SousChoix eq "delrep") {
    # SUPPRIMER UN ATTRIBUT DE RÉPONSE A UN GROUPE
    # Récupérer les informations en mode Intéractif, sinon en mode Automatique utiliser les arguments de la commande
    if (!$Group) {
          print "Groupe : " ;
          $Group = <STDIN> ;
          chomp $Group ;
    }
    if (!$Attribute) {
          print "\nAttribut : " ;
          $Attribute = <STDIN> ;
          chomp $Attribute ;
    }
    if (!$reponse) {
          # Demander une confirmation avant de faire l'irrémédiable.
          print "Confirmer suppression ? (o/n)" ;
          $reponse = <STDIN> ;
          chomp $reponse ;
    }
    # Réaliser la suppression.
    if ($reponse eq "o") {
          $ResReq = $dbh->do("DELETE FROM radgroupreply WHERE GroupName=\"$Group\" AND Attribute=\"$Attribute\" ;") ;
          print "[OK] Suppression effectuée\n" ;
          sleep 1 ;
    }
    else {
          # Ne rien faire
          print "\n[NOK] Suppression non-réalisée (action suspendue par l'utilisateur)\n" ;
    }
} # Fin du if ($SousChoix eq "26" or $SousChoix eq "delrep")

if ($SousChoix eq "27" or $SousChoix eq "delgroup") {
    # SUPPRIMER UN GROUPE ET TOUS SES ATTRIBUTS
    # Récupérer les informations en mode Intéractif, sinon en mode Automatique utiliser les arguments de la commande
    if (!$Group) {
          print "Groupe : " ;
          $Group = <STDIN> ;
          chomp $Group ;
    }
    if (!$MdP) {
          print "\nMot de passe de l'utilisateur de la base : " ;
          $MdP = <STDIN> ;
          chomp $MdP ;
    }
```

```
            if (!$reponse) {
                  # Demander une confirmation avant de faire l'irrémédiable.
                  print "Confirmer suppression ? (o/n)" ;
                  $reponse = <STDIN> ;
                  chomp $reponse ;
            }

            # Il faut le mot de passe user valide et la confirmation de la suppression pour l'effectuer.
            if (($reponse eq "o") and ($MdP eq $PasswordDB)) {
                  # Supprimer le groupe dans la base de données.
                  $ResReq = $dbh->do ("DELETE FROM radgroupcheck WHERE GroupName LIKE \"$Group\" ;") ;
                  $ResReq = $dbh->do ("DELETE FROM radgroupreply WHERE GroupName LIKE \"$Group\" ;") ;
                  $ResReq = $dbh->do ("DELETE FROM usergroup WHERE GroupName LIKE \"$Group\" ;") ;
                  print "\n[OK] Suppression effectuée\n" ;
            }
            else {
                  # Ne rien faire.
                  print "\n[NOK] Suppression non-réalisée (mauvais mot de passe ou action suspendue par l'utilisateur)\n" ;
            }
      } # Fin if ($SousChoix eq "27" or $SousChoix eq "delgroup")

   } # Fin if ($Choix eq "2")

   if ($Choix eq "3" or $Choix eq "nas") {
         # Affichage du menu si l'on n'est pas en mode Automatique.
         if (!$Option) {
             print "\n\n##############################################\n" ;
             print "31 - Ajouter un équipement réseau\n" ;
             print "32 - Supprimer un équipement réseau\n" ;
             print "33 - Lister un ou tous les équipements réseaux\n" ;
             print "##############################################\n" ;

             # Lecture du choix de l'utilisateur en mode Intéractif
             print "Choix : " ;
             $SousChoix = <STDIN> ;
             chomp $SousChoix ;
         }
         if ($SousChoix eq "31" or $SousChoix eq "add") {
             # AJOUTER UN NAS
             # Récupérer les informations en mode Intéractif, sinon en mode Automatique utiliser les arguments de la commande
             if (!$Nasname) {
                   print "Adresse IP de l'équipement réseau : " ;
                   $Nasname = <STDIN> ;
                   chomp $Nasname ;
             }
             if (!$Shortname) {
                   print "\nNom court : " ;
                   $Shortname = <STDIN> ;
                   chomp $Shortname ;
             }
             if (!$Type) {
                   print "\nType d'équipement réseau : " ;
                   $Type = <STDIN> ;
                   chomp $Type ;
             }
             if (!$Ports || $Ports != 0) {
                   print "\nPort : " ;
                   $Ports = <STDIN> ;
                   chomp $Ports ;
             }
             if (!$Secret) {
                   print "\nMot de passe partagé : " ;
                   $Secret = <STDIN> ;
                   chomp $Secret ;
             }
             if (!$Group) {
                   print "\nGroupe : " ;
                   $Group = <STDIN> ;
                   chomp $Group ;
             }
             if (!$Description) {
                   print "\nDescription ou commentaire : " ;
                   $Description = <STDIN> ;
                   chomp $Description ;
             }

             # Rajouter un NAS dans la base de donénes après avoir vérifié qu'il n'existe pas déjà.
             # .. Vérification sur le nas.nasname
             $ResReq = $dbh->prepare("SELECT nasname FROM nas WHERE nasname LIKE \"$Nasname\" and groupe LIKE \"$Group\" ;") ;
             $ResReq->execute() ;
             $ResReqLign = $ResReq->fetchrow_hashref() ;
             if ($ResReqLign->{'nasname'} ne "") {
                   # Ne rien faire car le nas existe déjà.
                   print "[NOK] L'équipement réseau existe déjà pour ce groupe\n" ;
                   $ResReq->finish() ;
             }
             else {
                   # Nettoyer la requête et faire l'insertion.
                   $ResReq->finish() ;
                   $ResReq = $dbh->do("INSERT INTO nas (nasname, shortname, type, ports, secret, description, groupe) VALUES
(\"$Nasname\", \"$Shortname\", \"$Type\", $Ports, \"$Secret\", \"$Description\", \"$Group\") ;") ;
                   # Si cet équipement n'a pas été affecté à admin, alors le rajouter
                   $ResReqInterm = $dbh->prepare("SELECT * FROM nas WHERE nasname LIKE \"$Nasname\" and groupe LIKE \"admin\" ;") ;
                   $ResReqInterm->execute() ;
                   if (! $ResReqInterm->fetchrow_hashref()) {
                       $ResReq = $dbh->do("INSERT INTO nas (nasname, shortname, type, ports, secret, description, groupe) VALUES
(\"$Nasname\", \"$Shortname\", \"$Type\", $Ports, \"$Secret\", \"$Description\", \"admin\") ;") ;
                       print "[OK] Insertion effectuée avec ajout du groupe admin.\n" ;
                   }
                   else {
                       print "[OK] Insertion effectuée.\n" ;
                   }
                   $ResReqInterm->finish() ;
                   # Si on n'est pas en mode automatique, on redémarre FreeRADIUS à chaque modification.
                   if (!$Option) {
                       print "Redémarrage de FreeRADIUS pour prise en compte.\n" ;
                       sleep 1 ;
                       print `/etc/init.d/freeradius restart` ;
                   }
             }

         } # Fin if ($SousChoix "31")

         if ($SousChoix eq "32" or $SousChoix eq "del") {
             # SUPPRIMER UN NAS
             # Récupérer les informations en mode Intéractif, sinon en mode Automatique utiliser les arguments de la commande
             if (!$Nasname) {
                   print "Adresse IP de l'équipement réseau : " ;
```

```
                $Nasname = <STDIN> ;
                chomp $Nasname ;
            }
            if (!$MdP) {
                print "\nMot de passe de l'utilisateur de la base : " ;
                $MdP = <STDIN> ;
                chomp $MdP ;

                # Demander une confirmation avant de faire l'irrémédiable.
                print "Confirmer suppression ? (o/n)" ;
                $reponse = <STDIN> ;
                chomp $reponse ;
            }

            # Il faut le mot de passe user valide et la confirmation de la suppresion pour l'effectuer.
            if (($reponse eq "o") and ($MdP eq $PasswordDB)) {
                # Suppremier le nas dans la base de données.
                $ResReq = $dbh->do ("DELETE FROM nas WHERE nasname LIKE \"$Nasname\" ;") ;
                print "\n[OK] Suppression effectuée. Redémarrage de FreeRADIUS pour prise en compte.\n" ;
                sleep 1 ;
                print `/etc/init.d/freeradius restart` ;

            }
            else {
                # Ne rien faire.
                print "\nSuppression non-réalisée (mauvais mot de passe ou action suspendue par l'utilisateur)\n" ;
            }
            sleep 1 ;
        } # Fin if ($SousChoix eq "32")

        if ($SousChoix eq "33" or $SousChoix eq "list") {
            # LISTER LE NAS DÉFINI SINON LISTER TOUS LES NAS.
            # .. Demande du nas à afficher si l'on n'est pas en mode Automatique.
            if (!$Option) {
                print "Nom (laisser vide pour tout afficher) : " ;
                $Nom = <STDIN> ;
                chomp $Nom ;
            }
            # .. Si on a une adresse IP, on ne liste pas tout.
            if ($Nom =~ /\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/) {
                $ResReq = $dbh->prepare("SELECT * FROM nas WHERE nasname LIKE \"$Nom\" ORDER by id ;") ;
            }
            else {
                $ResReq = $dbh->prepare("SELECT * FROM nas ORDER by id ;") ;
            }
            $ResReq->execute() ;
            # Ligne de titre
            print "-------------------------------------------------------------------------------------------------------------------------\n" ;
            print "|      Nom       |   Nom court    |  Type  |Ports|     Secret     |   Groupe    |Communauté|      Description     |\n" ;
            print "-------------------------------------------------------------------------------------------------------------------------\n" ;
            # Résultat
            $~ = "ListNas" ;
            while ($ResReqLign = $ResReq->fetchrow_hashref()) {
                write() ;
            }
            print "-------------------------------------------------------------------------------------------------------------------------\n" ;
            # Nettoyer la requête
            $ResReq->finish() ;
        } # Fin if ($SousChoix eq "33")

    } # Fin if ($Choix eq "3")


    if ($Choix eq "4" or $Choix eq "adm") {
        # Affichage du menu si l'on n'est pas en mode Automatique.
        if (!$Option) {
            print "\n\n##############################################\n" ;
            print "41 - Synchroniser les BD du serveur principal vers le secours\n" ;
            print "42 - Lancer une sauvegarde\n" ;
            print "43 - Afficher les tentatives de connexion erronées\n" ;
            print "44 - Afficher les tentatives de connexion réussies\n" ;
            print "45 - Afficher les dernières connexions au système\n" ;
            print "46 - Afficher l'état des serveurs\n" ;
            print "47 - Watchdog (services lancés ?)\n" ;
            print "##############################################\n" ;

            # Lecture du choix de l'utilisateur en mode Intéractif
            print "Choix : " ;
            $SousChoix = <STDIN> ;
            chomp $SousChoix ;
        }
        if ($SousChoix eq "41" or $SousChoix eq "sync") {
            # SYNCHRONISER les BD DU SERVEUR PRINCIPAL VERS LE SERVEUR DE SECOURS
            # dump de la base injecté sur le serveur distant (par tunnel ssh).
            print `mysqldump -h localhost -P 3306 -u $LoginDB -p$PasswordDB $DatabaseDB | mysql -h 127.0.0.1 -P 30000 -u $LoginDB -p$PasswordDB
$DatabaseDB` ;
            print "Sauf si erreur ci-dessus, base du serveur de secours mise à jour par rapport à la base du serveur principal.\n" ;
        } # Fin if ($SousChoix eq "41")

        if ($SousChoix eq "42" or $SousChoix eq "sauv") {
            # LANCER UNE SAUVEGARDE
            $DateAnnee = localtime->year + 1900 ;
            $DateMois = localtime->mon + 1 ;
            $DateJour = localtime->mday ;
            $DateHeure = localtime->hour ;
            $DateMinute = localtime->min ;
            $DateSeconde = localtime->sec ;
            $NomBase = "RADIUS".$DateAnnee."_".$DateMois."_".$DateJour."_".$DateHeure."_".$DateMinute."_".$DateSeconde ;

            # Récupérer les informations en mode Intéractif, sinon en mode Automatique utiliser les arguments de la commande
            if (!$LoginDB) {
                print "Veuillez indiquer le login administrateur de la base de données : " ;
                $LoginDB = <STDIN> ;
                chomp $LoginDB ;
            }
            if (!$PasswordDB) {
                print "Veuillez indiquer le mot de passe administrateur de la base de données : " ;
                $PasswordDB = <STDIN> ;
                chomp $PasswordDB ;
            }
            # dump de la base dans un fichier et dans une base de sauvegarde.
            print `mysqldump -h localhost -P 3306 -u $LoginDB -p$PasswordDB $DatabaseDB > /adminradius/FichierSauvegarde_$NomBase` ;

            # Vérification que les tables radcheck, nas et usergroup ne sont pas vides dans la table de sauvegarde.
            ## Préparation des requêtes sur la base de production et sur la base de sauvegarde.
            $ResReq = $dbh->prepare("SELECT count(*) FROM radcheck ;") ;
            $ResReqBis = $dbh->prepare("SELECT count(*) FROM nas ;") ;
            $ResReqTri = $dbh->prepare("SELECT count(*) FROM usergroup ;") ;
```

```
        ## Exécution des requêtes.
        $ResReq->execute() ;
        $ResReqBis->execute() ;
        $ResReqTri->execute() ;

        ## Récupération des résultats.
        $ResReqLign = $ResReq->fetchrow_hashref() ;
        $ResReqBisLign = $ResReqBis->fetchrow_hashref() ;
        $ResReqTriLign = $ResReqTri->fetchrow_hashref() ;


        ## Vérification et affichage du résultat de la vérification.
        print "\n" ;
        $ComptPb = 0 ;
        if ($ResReqLign->{'count(*)'} > 0) {
              print "Vérification de la table radcheck sur base de production : OK\n" ;
        }
        else {
              print "Vérification de la table radcheck sur base de production : NOK, pb sauvegarde !\n" ;
              $ComptPb ++ ;
        }
        if ($ResReqBisLign->{'count(*)'} > 0) {
              print "Vérification de la table nas sur base de production : OK\n" ;
        }
        else {
              print "Vérification de la table nas sur base de production : NOK, pb sauvegarde !\n" ;
              $ComptPb ++ ;
        }
        if ($ResReqTriLign->{'count(*)'} > 0) {
              print "Vérification de la table usergroup sur base de production : OK\n" ;
        }
        else {
              print "Vérification de la table usergroup sur base de production : NOK, pb sauvegarde !\n" ;
              $ComptPb ++ ;
        }
        if (-e "/adminradius/FichierSauvegarde_$NomBase") {
              print "Vérification existance du fichier de sauvegarde : OK\n" ;
        }
        else {
              print "Vérification existance du fichier de sauvegarde : NOK, pb sauvegarde !\n" ;
              $ComptPb ++ ;
        }

        if ($ComptPb != 0) {
              print "\n[NOK] La sauvegarde ne s'est pas correctement réalisée\n" ;
        }
        else {
              print "\n[OK] La sauvegarde s'est correctement déroulée\n" ;
        }

        # Nettoyage des requêtes
        $ResReq->finish() ;
        $ResReqBis->finish() ;
        $ResReqTri->finish() ;
    } # Fin if ($SousChoix eq "42")

    if ($SousChoix eq "43" or $SousChoix eq "listerr") {
        # AFFICHER LES TENTATIVES DE CONNEXION ERRONEES
        # Ouvrir le fichier de log.
        open (LOGFREERADIUS, '/var/log/freeradius/radius.log') || die "Problème à l'accès au fichier /var/log/freeradius/radius.log" ;
        # Tant qu'il y a une ligne à lire, si elle a le bon format, l'afficher.
        while ($ligneFichierLog = <LOGFREERADIUS>) {
#             if (($ligneFichierLogOK) = $ligneFichierLog =~ /(.*No matching entry in the database for request from user.*)/) {
              if (($ligneFichierLogOK) = $ligneFichierLog =~ /(.*Auth: Login incorrect.*)/) {
                  print "$ligneFichierLogOK\n" ;
              }
        }
        # Fermer le fichier de log.
        close (LOGFREERADIUS) ;
    } # Fin if ($SousChoix eq "43")

    if ($SousChoix eq "44" or $SousChoix eq "listok") {
        # AFFICHER LES TENTATIVES DE CONNEXION REUSSIES
        # En mode non Automatique, utiliser les arguments de la commande
        # Affichage de la table radpostauthsous forme d'un tableau.
        $ResReq = $dbh->prepare("SELECT * FROM radpostauth ;") ;
        $ResReq->execute() ;
        print "-------------------------------------------------------------------------------\n" ;
        print "|  id  |      user       |         reply        |         date         |\n" ;
        print "-------------------------------------------------------------------------------\n" ;
        $~ = "ListConnOK" ;
        while ($ResReqLign = $ResReq->fetchrow_hashref()) {
              write() ;
        }
        print "-------------------------------------------------------------------------------\n" ;
        $ResReq->finish() ;
        sleep 1 ;
    } # Fin if ($SousChoix eq "44")

    if ($SousChoix eq "45" or $SousChoix eq "last") {
        # AFFICHER LES DERNIÈRES CONNEXIONS AU SYSTÈME
        print `last` ;
        sleep 1 ;
    } # Fin if ($SousChoix eq "45")

    if ($SousChoix eq "46" or $SousChoix eq "state") {
        # AFFICHER UN ETAT DES SERVEURS
        print "Appuyer sur ENTREE quand vous voudrez passer d'une rubrique à l'autre...\n" ;
        sleep 2 ;

        print "Uptime du serveur principal :\n" ;
        print `uptime` ;
        if (!$Option) {
              $Entree = <STDIN> ;
        }
        print "\n\nEspace disque du serveur principal : \n" ;
        print `df` ;
        if (!$Option) {
              $Entree = <STDIN> ;
        }
        print "\n\nUtilisation de la swap du serveur principal :\n" ;
        print `free`;
        if (!$Option) {
              $Entree = <STDIN> ;
        }
        print `netstat -i` ; # Information sur les interfaces réseaux
        if (!$Option) {
```

```
                $Entree = <STDIN> ;
            }
            print `netstat -r` ; # Information sur la table de routage
            if (!$Option) {
                $Entree = <STDIN> ;
            }
            print `netstat -s` ; # Information sur les stats des couches réseaux (cf SNMP)
            if (!$Option) {
                $Entree = <STDIN> ;
            }
            print `netstat -ople` ; # Information sur les sockets avec timers, programme et owner.
            if (!$Option) {
                $Entree = <STDIN> ;
            }

            print "S'il est nécessaire de faire la même analyse sur le serveur de secours, soit se connecter sur celui-ci et taper la ligne de
commande :\n" ;
            print "uptime && sleep 10 && df && sleep 10 && free && sleep 10 && netstat -i && sleep 10 && netstat -r && sleep 10 && netstat -s && sleep 10
&& netstat -ople\n" ;
            print "soit depuis le serveur principal taper la commande suivante (avec les bonnes valeurs) :\n" ;
            print "ssh <login>\@<adresseIPServeurSecours> \"uptime && df && free && netstat -i && netstat -r && netstat -s && netstat -ople\"\n" ;
            sleep 1 ;
        } # Fin if ($SousChoix eq "46")

        if ($SousChoix eq "47" or $SousChoix eq "watch") {
            # VERIFIER QUE LES SERVICES SONT ACTIFS
            $ErreurService = 0 ;
            # Vérification de FreeRADIUS.
            ## Est-ce que le ficher pid existe ?
            if (-e "/var/run/freeradius/freeradius.pid") {
                ## Est-ce que le processus indiqué existe ?
                open (PIDFILE, "/var/run/freeradius/freeradius.pid") ;
                $pid = <PIDFILE> ;
                chomp $pid ;
                close (PIDFILE) ;
                if (-e "/proc/$pid") {
                    print "[OK] FreeRADIUS est lancé\n" ;
                }
                else {
                    print "[NOK] FreeRADIUS planté\n" ;
                }
            }
            else {
                print "[NOK] FreeRADIUS non lancé\n" ;
                $ErreurService = $ErreurService + 1 ;
            }
            # Vérification de syslogd.
            ## Est-ce que le fichier pid existe ?
            if (-e "/var/run/syslogd.pid") {
                ## Est-ce que le processus indiqué existe ?
                open (PIDFILE, "/var/run/syslogd.pid") ;
                $pid = <PIDFILE> ;
                chomp $pid ;
                close (PIDFILE) ;
                if (-e "/proc/$pid") {
                    print "[OK] Syslogd est lancé\n" ;
                }
                else {
                    print "[NOK] Syslogd planté\n" ;
                }
            }
            else {
                print "[NOK] Syslogd non lancé" ;
                $ErreurService = $ErreurService + 1 ;
            }
            # Vérification de mysqld.
            ## Est-ce que le fichier pid existe ?
            if (-e "/var/run/mysqld/mysqld.pid") {
                ## Est-ce que le processus indiqué existe ?
                open (PIDFILE, "/var/run/mysqld/mysqld.pid") ;
                $pid = <PIDFILE> ;
                chomp $pid ;
                close (PIDFILE) ;
                if (-e "/proc/$pid") {
                    print "[OK] MySQL est lancé\n" ;
                }
                else {
                    print "[NOK] MySQL est planté\n" ;
                }
            }
            else {
                print "[NOK] MySQL non lancé\n" ;
                $ErreurService = $ErreurService + 1 ;
            }
            # Vérification de sshd.
            ## Est-ce que le fichier pid existe ?
            if (-e "/var/run/sshd.pid") {
                ## Est-ce que le processus indiqué existe ?
                open (PIDFILE, "/var/run/sshd.pid") ;
                $pid = <PIDFILE> ;
                chomp $pid ;
                close (PIDFILE) ;
                if (-e "/proc/$pid") {
                    print "[OK] SSHD est lancé\n" ;
                }
                else {
                    print "[NOK] SSHD est planté\n" ;
                }
            }
            else {
                print "[NOK] SSHD non lancé\n" ;
                $ErreurService = $ErreurService + 1 ;
            }
            # Vérification de openntpd.
            print "[WA] Openntpd ne peut être vérifié\n" ;
            # Synthèse
            if ($ErreurService != 0) {
                print "[NOK] Déclenchement du watchdog\n" ;
                sleep 2 ;
                open(WATCHDOG, "/dev/watchdog") || die "Pb ouverture /dev/watchdog. Le module softdog est-il chargé ?" ;
                close(WATCHDOG) ;
            }
        } # Fin if ($SousChoix eq "47" or $SousChoix eq "watch")

    } # Fin if ($Choix eq "4")

    # Purge de toutes les variables.
    $Nom = "" ;
```

```
    $MdP = "" ;
    $Group = "" ;
    $NewMdP = "" ;
    $Attribute = "" ;
    $Op = "" ;
    $Value = "" ;
    $Nasname = "" ;
    $Shortname = "" ;
    $Type = "" ;
    $Ports = "" ;
    $Secret = "" ;
    $Description = "" ;
    $LoginRootDB = "" ;
    $PasswordRootDB = "" ;

    # Pour ne pas boucler indéfiniement en mode Automatique et sortir une fois l'action réalisée ...
    if ($Option == 1) {
         $Choix = 0 ;
    }

} # fin du while ($Choix ne "0")

# Fermeture de la base de données
$dbh->disconnect() ;
```

Annexe : Code source de ChgtMdP.pl

```
#!/usr/bin/perl

# v0.8 Programme de Guillaume Lehmann. Programme protégé par la licence GNU Public Licence version 3.

#################################################################
# SCRIPT POUR LE CHANGEMENT DE MOT DE PASSE SYSTEME ET RADIUS #
# POUR LES UTILISATEURS                                         #
# Il faut s'être connecté avec son propre compte et connaître #
# le mot de passe RADIUS pour pouvoir ensuite changer le mot  #
# de passe                                                      #
#################################################################

use DBI ;


# Message d'accueil
$Utilisateur = getlogin() ;
print "Bonjour vous êtes l'utilisateur $Utilisateur\n" ;


# Utiliser la ligne ci-dessous plutôt si le programme n'est pas une fonction.
#my ($Utilisateur, $mdpPlain) = @ARGV ;

# Utiliser la ligne ci-dessous plutôt si le programme est dans une fonction.
# my ($Utilisateur, $mdpPlain) = @_ ;

# Changement du mot de passe système.
print "Changement du mot de passe du système ... \n" ;
$ResultProg = system (passwd) ;
if ($ResultProg != 0) {
    print "Erreur format mot de passe, merci de retenter une connexion\n" ;
    sleep 2 ;
    die "Erreur lors du changement de mot de passe" ;
}

# Changement du mot de passe sous FreeRADIUS.
print "Passons maintenant au changement de mot de passe sous FreeRADIUS\n" ;

# Ouverture de la base de données.
open(ACCESBD, '/adminradius/AccesBD.txt') || die "Problème à  l'accès au fichier AccesBD.txt" ;
$resultat = <ACCESBD> ;
close (ACCESBD) ;
($database, $hostname, $login, $password, $typBD) = $resultat =~ /^(\w*)\s*(\w*)\s*(\w*)\s*(\w*)\s*(\w*)\s*$/ ;
$dsn = "DBI:$typBD:database=$database;host=$hostname" ;

$dbh = DBI->connect($dsn, $login, $password) ;

# Demande des mots de passe.
print "Veuillez entrer votre nouveau mot de passe RADIUS : " ;
$MdPPlain1 = <STDIN> ;
chomp $MdPPlain1 ;
if ($MdPPlain1 !~ /[a-z,A-Z,\_,0-9]/) {
    print "Erreur format mot de passe, merci de retenter une connexion\n" ;
    sleep 2 ;
    die "Erreur format mot de passe" ;
}


print "Veuillez le confirmer : " ;
$MdPPlain2 = <STDIN> ;
chomp $MdPPlain2 ;
if ($MdPPlain2 !~ /[a-z,A-Z,\_,0-9]/) {
    print "Erreur format mot de passe, merci de retenter une connexion\n" ;
    sleep 2 ;
    die "Erreur format mot de passe" ;
}

# Vérification de l'identité de la personne.
# $Req = $dbh->prepare("SELECT * FROM radcheck WHERE UserName LIKE \"$Utilisateur\" AND Attribute LIKE \"Password\" AND Value
LIKE \"$MdPCurrent\" ;") ;
$Req = $dbh->prepare("SELECT * FROM radcheck WHERE UserName LIKE \"$Utilisateur\" ;") ;
$Req->execute() ;


# Si le login correspond on continue sinon on indique un message d'erreur et programme terminé.
```

```
if ($ResReqLign = $Req->fetchrow_hashref()) {
    $Req->finish() ;
    # Si les 2 mots de passe saisis sont les même alors on fait le changement, sinon on sort.
    if ($MdPPlain1 eq $MdPPlain2) {
        # Création des paramètres de création du mot de passe système.
        $chaine = './0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
        # Création de la graine.
        for ($i = 0; $i < 8; $i++) {
            $graine .= substr($chaine,
                              rand(length($chaine)), 1);
        }
        # Création du mot de passe chiffré au format MD5 (donc graine au format $1$ suivi de 8 caractères puis $) .
        $mdpCrypt = crypt($MdPPlain1, '$1$'.$graine.'$');


        $dbh->do("UPDATE radcheck SET Value = \"$mdpCrypt\", Attribute = \"Crypt-Password\" WHERE UserName LIKE \"$Utilisateur\" ;")
;
        print "[OK] Changement du mot de passe radius effectué. Sortie de l'utilitaire dans 5 secondes\n\n" ;
        sleep 5 ;
    }
    else {
        print "[NOK] Les mots de passes sont différents. Sortie de l'utilitaire dans 5 secondes\n" ;
        sleep 5 ;
    }

} # Fin "if ($ResReqLign = $ResReq->fetchrow_hashref())"
else {
    print "[NOK] Le mot de passe courant et/ou le login indiqué est erronné. Sortie de l'utilitaire dans 5 secondes\n" ;
    sleep 5 ;
}
$Req->finish() ;
$dbh->disconnect() ;
```

Annexe : Code source de chroot

```
#!/bin/bash
# Script v0.3 de Guillaume Lehmann

# Mettre /root/chroot comme shell des utilisateurs dans le fichier /etc/passwd .
# Si on veut "chrooter" le compte et y exécuter un script perl au lieu du bash (créer l'environnement chroot avant !)
#exec -c /usr/sbin/chroot /home/$USER /usr/bin/perl MonScriptPerl.pl

# Si on veut exécuter le script perl /adminradius/ChgtMdP.pl sans chroot
exec -c /usr/bin/perl /adminradius/ChgtMdP.pl

# Si on veut juste exécuter /bin/bash comme si on l'avait mis dans /etc/passwd .
#/bin/bash
```

Annexe : Contenu de AccesBD.txt

Le fichier est composé d'une ligne, par exemple comme suit :

```
radius localhost radiusXXX radiusXXXMdP mysql
```

C'est une suite de 5 mots séparés par un ou plusieurs espaces (ou tabulations). Le premier mot indique le nom de la base, le second l'hôte hébergeant la base de données, les troisième et quatrième mots sont utilisés respectivement comme login et mot de passe pour se connecter à la base de donnée. Enfin, le cinquième indique le type de SGBD. Dans notre exemple, nous nous connectons à la base radius hébergée en local. Pour cela, nous utilisons le login radiusXXX et le mot de passe radiusXXXMdP. Pour finir, le SGBD est mysql (écrit en minuscule).

Annexe : Récapitulatif des utilisateurs déclarés sur le système

Voici un récapitulatif des utilisateurs déclarés dans le système ainsi que les mots de passe indiqués dans la présente documentation. Il faut absolument changer ces mots de passe qui ne sont indiqués qu'à titre d'exemple !

Administrateur système :

- Login : root
- Mot de passe : <mot de passe indiqué lors de l'installation système>

Utilisateur pour lancer FreeRADIUS :

- Login : freerad
- Mot de passe : ne peut pas se connecter (*/bin/false*)

Utilisateur pour lancer MySQL :

- Login : mysql
- Mot de passe : ne peut pas se connecter (*/bin/false*)

Utilisateur ayant des accès normaux à MySQL (base radius) :

- Login : radiusXXX
- Depuis : @localhost
- Mot de passe : radiusXXXMdP

Utilisateur ayant tous les droits dans MySQL :

- Login : root
- Depuis : @localhost
- Mot de passe : MotDePasseRoot

Utilisateur ayant les accès administrateur du système et ayant la console de supervision :

- Login : adminradius
- Mot de passe : <mot de passe indiqué lors de l'installation système>

Utilisateur pour réception du tunnel ssh sur le serveur de secours :

- Login : vpnssh
- Mot de passe : <mot de passe indiqué lors de l'installation système>

Annexe : Choisir un bon mot de passe

2 types d'attaques existent pour les mots de passe. La première par force brute, la seconde par statistique.
En calculant le temps mis pour casser un mot de passe par force brute on obtient la limite supérieure du délai théorique pour casser le mot de passe. L'attaque par statistique est généralement plus rapide que la force brute.

Pour calculer la force d'un mot de passe :

- Si un mot de passe ne contient que des majuscules ou que des minuscules, le nombre de combinaisons possibles s'étend à 26 à la puissance N où N est le nombre de lettres.
- Si un mot de passe ne contient que des chiffres, le nombre de combinaisons possibles s'étend à 10 à la puissance N où N est le nombre de chiffres.
- Si un mot de passe contient des majuscules et des minuscules le nombre de combinaisons possibles s'étend à 52 à la puissance N, et en ajoutant des chiffres on monte à 62 à la puissance N.

Si on part du principe qu'un PC personnel peut tester 100 millions de mots de passe à la seconde (ce qui est très élevé), alors on peut calculer combien de temps il faudrait pour trouver le mot de passe choisi.
Pour tuy7Yy0p, il faudra au plus 2183401 secondes pour le trouver, soit entre 25 et 26 jours.
En rajoutant une seule lettre/chiffre, il faudra alors au plus 135370865 secondes pour le trouver, soit entre 1566 et 1567 jours !
En réalité, si l'attaque est linéaire, tuy7Yy0p sera trouvée en environ 189h soit moins de 8 jours, et rajouter une lettre majuscule (lettre K) prolongera le suspens de plus de 5340 heures, soit 222 jours de plus …

Maintenant à vous de choisir vos mots de passe, mais sachez que les mots de passe indiqués dans cette documentation sont donnés pour illustrer les commandes et les explications, non comme référence en matière de sécurité.