http://clx.asso.fr/spip/?Utiliser-Linux-oui-mais-pas-les

Utiliser Linux... oui mais pas les yeux fermés !

- Documentations - Installation / Administration de base de Linux -

Date de mise en ligne : mardi 5 novembre 2002

Cet article a pour vocation de protéger un minimum le systÚme d'exploitation Linux avec des notions simples, valables quelles que soient les distributions. Il s'adresse essentiellement aux non-informaticiens.

Buts Objectifs

Chaque minutes de chaque jour, des centaines de réseaux connectés à Internet sont soumis à des attaques par des programmes de recherche automatique afin de trouver des vulnérabilités et les exploiter. Voilà comment commence un article trouvé sur linuxsecurity.com. Votre PC Linux est connecté à Internet ? il est donc soumis à ces possibilités d'attaques avec des outils logiciels automatiques qui vont exploiter les failles qu'ils trouvent. Cet article est destiné à vous faire prendre conscience de ces risques, et proposer un ensemble de mesures destinées à mettre en place, ou améliorer le niveau de sécurité de base sur un système Linux connecté en réseau.

Nombre d'utilisateurs, issus du monde Windows ou Mac, ont une culture informatique qui ne les prédispose pas toujours à connaître les fonctionnalités et arcanes d'un système d'exploitation multi-utilisateur et multitâches (i.e. plusieurs programmes s'exécutent de façon quasiment simultanée)... En outre la connexion de nos postes à un réseau planétaire comme Internet change les données du problème par rapport aux années 80. L'ouverture sur un réseau de cette échelle centuple les risques d'actes de malveillance informatique : intrusions, ou refus de services. En effet sur un système multitâche comme Linux, un grand nombre de programmes sont présents et actifs en mémoire. Appelés *serveurs* ou *demons*, ils s'exécutent sans trace apparente à l'écran (un serveur de messagerie, un serveur www par exemple). Cependant sur votre machine, ces programmes sont à l'écoute, et en attente de connexions provenant de l'extérieur. Si le poste Linux est peu ou mal administré, trop ouvert sur l'extérieur, ou encore s'il propose des versions de logiciels obsolètes, les facteurs de risques permettant des actes malveillants seront d'autant plus grands. Ces mises en garde générales ne sont d'ailleurs pas l'apanage de Linux seulement, mais concernent tous les systèmes d'exploitation multitâches.

Linux posséde de base une grande variété de programmes pouvant assurer de la sécurité.

Ces lignes visent à :

– attirer l'attention de certains utilisateurs sur la nécessité de mettre en place une administration de base de la machine prenant en compte la sécurité du sytème et du réseau
– faire connaître et passer en revue quelques éléments pour mettre en place un niveau de sécurité de base.

Lors de l'Installation

Certaines distributions Linux (Redhat, Mandrake, Suse, Debian..) facilitent la phase d'installation du système en proposant des classes d'installation prédéfinies. Dans ce cadre là, il s'agit de ne pas installer le système *à l'aveuglette*, sans connaître la portée de ce que l'installation automatique de votre distribution aura choisi pour vous. Il est en outre nécessaire de bien réfléchir à quoi va servir la machine que l'on installe ? est-ce une station de travail de bureautique, de développement ? ou bien une machine serveur réseau ?.. Par exemple une distribution Linux

Mandrake propose des classes d'installation telles que :

– station de travail : classe d'installation la plus polyvalente. Celle ci installe un ensemble de programmes standards (bureautique, clients internet, logiciels scientifiques...)

– développement : Pour les utilisateurs qui souhaitent développer des logiciels ou les compiler. Le système installe un grand nombre de librairies de sous programmes (`*.a, *.so`) et de fichiers d'entêtes (`*.h`) nécessaires au développement d'applications et la compilation de programmes en "C" ou autre langage (Perl, Python...).

– serveur : classe permettant de transformer votre machine en serveur réseau. Un grand nombre de serveurs réseau sont proposés et installés (dns, mail, ftp, web, news,...).

D'autres distributions (Redhat) proposent l'installation d'ensembles de paquetages homogènes (Serveurs réseau, jeux, bureautique, bureaux...). Dans tous les cas, il sera nécessaire de :

– vérifier et compléter les choix prédéfinis en choisissant l'option **choix des paquetages individuellement**. Cela vous permettra de découvrir et apprendre à connaître les noms et fonctions des différents paquetages logiciels issus du monde du **Logiciel Libre** afin de décider en pleine lumière si oui ou non ils sont nécessaires sur votre machine... Au bénéfice du doute, il est sûrement préférable de ne pas installer un service si on ne sait pas à quoi il sert. Il sera en effet très facile de le rajouter par la suite si on en a besoin à l'aide des commande `rpm` ou `dpkg`.
– choisir les services que l'on souhaite lancer au démarrage... Il est nécessaire de ne pas négliger cette phase et de bien choisir les services qui seront strictement nécessaires. Enfin, à l'issue de l'installation, il sera nécessaire de vérifier ce qui a été effectivement installé.

Post Installation : Vérifier les services lancés au démarrage

L'installation est terminée, félicitations ! Un certain nombre de paquetages sont donc installés sur la partition système Linux */*.

Il est maintenant nécessaire de passer en revue les services qui sont effectivement lancés en mémoire, afin de ne pas laisser de services inutiles qui pourraient être ultérieurement des cibles potentielles, pour d'éventuels pirates sur Internet.

Vérifions après l'installation, ce qui est vraiment lancé. Dans une fenêtre terminal, si vous tapez la commande :

```
$ chkconfig --level 3 --list
```

(ou bien chkconfig `--level 35 --list` pour inclure les niveaux de démarrage 3 et 5)

Vous verrez apparaître la liste de tous les services disponibles sur la machine avec l'indication de leur lancement au démarrage : "Arrêt" (ou Off) qui indique que le service n'est pas lancé au démarrage, et "Marche" (ou On) pour l'inverse. [Cette supervision s'obtient aussi en lancant certains programmes d'administration comme DrakConf sous Mandrake (choisir **services de démarrage**). Chaque distribution Linux a un outil d'administration graphique particulier pour vérifier cette liste des services démarrés. Nous préferons indiquer ici la commande de base `chkconfig` qui est commune à toutes les distributions.

```
... innd 0:Arrêt 1:Arrêt 2:Arrêt 3:Marche 4:Arrêt 5:Marche 6:Arrêt  ldap 0:Arrêt 1:Arrêt 2:Arrêt 3:Arrêt
```

```
4:Arrêt 5:Arrêt 6:Arrêt  nscd 0:Arrêt 1:Arrêt 2:Arrêt 3:Arrêt 4:Arrêt 5:Arrêt 6:Arrêt  sshd 0:Arrêt 1:Arrêt 2:Marche 3:Marche 4:Marche 5:Marche 6:Arrêt  ...
```

Ainsi par exemple : si je n'ai pas besoin du service *innd* (qui est un serveur de forums publics)

`$ rpm -qi innd` *pour avoir les informations sur ce paquetage*

```
$ rpm -e innd
```

pour désinstaller le paquetage

A contrario, si on veut lancer au démarrage un service qui ne l'était pas initialement, il suffit d'utiliser encore la commande `chkconfig` en indiquant successivement le niveau de démarrage, le nom du service et les mots clés "on" ou "off" (il faut être administrateur **root** pour lancer cette commande)

`$ chkconfig --level 35 webmin on` *lance par exemple le service webmin aux niveau 3 et 5 de démarrage*

Interdire le lancement de certaines applications

Que voit un pirate en herbe depuis l'extérieur de votre site ? Quels renseignements peut il obtenir sur votre machine ? Le logiciel nmap disponible sur Linux, est un outil d'exploration qui permet de déterminer quels services sont présents et actifs sur une machine connectée au réseau. Pour un éventuel pirate, il n'y a qu'à découvrir les services qu'une machine héberge, relever le numéro des versions des serveurs présents et tenter d'exploiter d'éventuelles failles (failles qui sont d'ailleurs décrites sur un grand nombre de sites Internet). Ces sites fournissent les vulnérabilités de telle ou telle application, ainsi que les programmes permettant d'exploiter cette vulnérabilité !.

Face à cette possibilité d'investigation par nmap, il est donc nécessaire :
– de montrer le moins de services ouverts possibles en désactivant ce qui est inutile, et
– de contrôler l'origine des connexions sur nos machines serveurs.

```
$ nmap monpc.monreseau.fr   Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ ) Interesting ports on localhost.localdomain (127.0.0.1):   (The 1507 ports scanned but not shown below are in state: closed)   Port State Service   21/tcp open ftp  22/tcp open ssh  25/tcp open smtp  80/tcp open http  110/tcp open pop-3 443/tcp open https  515/tcp open printer 6000/tcp open X11
```

On voit ici que la machine `monpc.monreseau.fr` offre un certain nombre de services disponibles depuis l'extérieur... on voit aussi que le service répondant sur le port 110 (POP) est une version QPopper en version 4.0.3

```
$ telnet monpc.monreseau.fr 110   Trying 194.254.147.252... Connected to monpc.monreseau.fr .  Escape character is '^]'.  +OK Qpopper (version 4.0.3) at monpc.monreseau.fr starting.
```

Désactiver les services inutiles

Le lancement des services réseau d'un système Linux repose sur un système dit **client-serveur**. Le schéma est simple : une machine **cliente** initie une connexion vers un **serveur** en demandant le lancement d'une application particulière. Ces services (encore appelé dans le jargon *démons*, ou *serveurs*) sont des programmes qui sont lancés de 2 manières différentes :

– soit par le programme inetd : inetd est un programme activé dès le démarrage du système, et qui attend en mémoire des demandes de connexion pour un service donné. Quand un client extérieur (ou même local à la machine) demande une connexion par le réseau, inetd détermine quel est le service à lancer en consultant son fichier de configuration `/etc/inetd.conf`.

– soit au démarrage de la machine : sans passer par inetd. C'est le cas de certains services très sollicités (comme un serveur de mail, ou un serveur www) qui doivent être présents en mémoire et répondre rapidement et directement aux demandes de connexion sans passer par le "super serveur" inetd.

Première étape de sécurité : vérifier le contenu de `/etc/inetd.conf`. Il ne faut laisser dans ce fichier que les services que l'on désire explicitement activer, et aucun autre (il suffit pour cela de placer un caractère de commentaire "#" devant la ligne du service que l'on souhaite désactiver.

```
# #pop-2   stream  tcp     nowait  root    /usr/sbin/tcpd ipop2d #pop-3   stream  tcp     nowait  root
/usr/sbin/tcpd /usr/local/sbin/popper -s  #spop3   stream  tcp     nowait  root    /usr/sbin/tcpd
/usr/local/sbin/popper -s -f /etc/mail/pop/qpopper.config #imap    stream  tcp     nowait  root
/usr/sbin/tcpd imapd #.
```

Deuxième étape de sécurité : ce schéma client-serveur basique est trop simple, et pas assez sécurisé ! En effet par défaut, on ne contrôle pas "qui" (i.e quelle machine cliente) est à l'origine de la demande. Or, les services qu'une machine propose n'ont pas forcément à être disponibles à tout l'Internet ! Il est nécessaire de restreindre l'utilisation des services réseau vers un ensemble de machines à qui on souhaite réserver ce service. Sauf cas particuliers de serveurs publics (comme un serveur WWW, ou de messagerie), les services réseau ne devraient pas être destinés à être accessibles depuis tout l'Internet. Aussi il faut sécuriser ce schéma de base simpliste de 2 manières :

– soit en utilisant tcpwrapper
– soit en utilisant xinetd

Contrôler l'accès aux services par tcpwrapper ou xinetd

Dans le contexte de sécurité actuel, l'utilisation de tcpwrapper ou xinetd sur Linux, devient indispensable ! L'idée de base de tcpwrapper (ou de xinetd comme successeur de inetd) est d'exercer un contrôle sur l'identité de la machine cliente qui veut lancer un service réseau du serveur. tcpwrapper permet donc de limiter les accès aux services réseau offerts par une machine serveur, en fonction de l'adresse réseau des machines clientes.

– utilisation de **tcpwrapper** : tcpwrapper est un petit programme appelé *tcpd* qui est lancé par inetd avant le service demandé par la machine cliente. Pour activer tcpwrapper (i.e tcpd) il suffit de le rajouter dans le fichier /etc/inetd.conf. Ce système est depuis longtemps intégré de base dans les distributions Linux (`rpm -q tcp_wrappers` pour vérifier sa présence).

```
 ftp stream tcp nowait root /usr/sbin/tcpd /opt/proftpd/sbin/proftpd
```

– Dans ce schéma, inetd lance tcpwrapper avant le service convoité, lequel tcpd va alors engager une série de vérifications pour savoir si le service demandé est permis ou non pour la machine cliente extérieure. les contrôles de tcpwrapper se font dans 2 fichiers seulement :

– `/etc/hosts.deny` : indique la listes des services qui sont interdits pour certaines machines et domaines internet. La

syntaxe est simple :

`<nom-du-service>:` (le mot clé "All" indique tous les services et/ou "tout le monde" selon l'endroit ou il est placé)

```
# ci dessous tous les services sont interdits à tout le monde All : All #
```

– `/etc/hosts.allow`

```
# autorisation de tous les services pour les machines de mon reseau ALL: LOCAL, .monreseau.fr  # autorisation d'accès à sshd pour tout le monde sshd: All  # acess au service POP que depuis les machines du reseau local poppassd: LOCAL, 192.168.1.0/255.255.255.0  popper: LOCAL, 192.168.1.0/255.255.255.0 # ## accès au serveur imap non sécurisé uniquement en LOCAL imapd: LOCAL, localhost.localdomain, 192.168.1.0/255.255.255.0 # ## accès au service imapS sécurisé par ssl pour tout le monde imapsd: ALL
```

Dans quel ordre se font les contrôles entre hosts.allow et hosts.deny :

– l'accès à un service est autorisé lorsque que tcpwrapper trouve un couple service:réseau dans le fichier hosts.allow,
– sinon le fichier hosts.deny est consulté et le service est refusé si une paire service:réseau s'y trouve.
– Sinon le service est accepté.

La politique la plus efficace dans ce fichier hosts.deny est de commencer par tout interdire en écrivant `All: All`. Ce qui signifie que tous les services de ma machine sont interdits à tout le monde. Cela nous permet de mettre en place une politique de sécurité visant à **Interdire tout ce qui n'est pas explicitement autorisé**. Il nous faut alors définir ce qui est explicitement autorisé dans /etc/hosts.allow

– **utilisation de xinetd** : en bref, xinetd est le programme qui est amené progressivement à remplacer inetd. xinetd intègre de façon native le supplément de sécurité de tcp_wrappers et apporte quelques éléments de contrôle supplémentaires. Quelques options supplémentaires et intéressantes de xinetd sont :

```
- disable = yes|no permet d'activer ou désactiver un service  - only_from = permet de restreindre le service à un ou plusieurs réseau  - no_access = interdit l'accès aux services pour les réseaux cités  - instances = "n" permet de contrôler la charge. xinetd ne lancera pas plus de "n" programmes demandés.  - access_times = 8:00-12:00 13:00-17:00 permet de restreindre l'utilisation d'un service sur une plage horaire.
```

On place ces options dans les fichiers de configuration présents dans /etc/xinetd.conf et /etc/xinetd.d :

```
service pop3 {     socket_type = stream     wait = no     user = root     server = /usr/sbin/ipop3d log_on_success += USERID     log_on_failure += USERID     disable = no    only_from = 192.168.1.0/24 }
```

On trouvera un très bon article en français sur les apports et nouveautés de xinetd sur :

http://www.fr.linuxfocus.org/Francais/November2000/article175.shtml

le problème des mots de passe

– **complexité** : On ne le dira jamais assez les mots de passe doivent être suffisamment complexes pour ne pas être trouvé par des individus, soit dotés de bon sens (pour trouver toto, azerty, vélo), soit dotés du logiciel *john-the-ripper*

qui permet de décrypter les mots de passe dans un fichier de mots de passe. En bref, un bon mot de passe doit comporter au minimum 6 caractères, être une combinaison de minuscules, majuscules, chiffres, et caractères de ponctuation. Le mot de passe ne doit pas être un mot du dictionnaire !. Le plus simple est de trouver une combinaison d'une phrase qui ait un sens mnémotechnique. Exemple : "j'ai 2 amours mon pays et Marseille" pourrait donner G2amPeM

– chiffrage : Il est bon de le savoir...ou de le rappeler : lorsque vous tapez quelque chose sur votre clavier, lors de la connexion à un service distant (un serveur pop3 par exemple, lorsque vous relevez vos mails)... tout ce que vous tapez, toutes les informations du dialogue client-serveur (et donc votre mot de passe), circulent par défaut de façon lisible (on dit *en clair* dans le jargon) sur le parcours entre le client et le serveur. Il suffit à un individu mal intentionné de lancer un programme d'écoute (un *sniffer* dans le jargon) sur un PC connecté au réseau, pour voir circuler vos noms et mots de passe de façon lisible.. imaginez la facilité de piratage par la suite !

Deux solutions principales s'offrent à nous pour contrecarrer ses possibilités d'écoute sur le réseau (en plus du fait d'avoir un mot de passe solide). Ces solutions sont progressivement mises en place par les administrateurs systèmes et réseaux de nos campus et laboratoires :

– avoir une architecture réseau la plus segmentée possible (en utilisant des commutateurs éthernet, ou des réseaux virtuels "vlan", par exemple)

– utiliser désormais des applications clients-serveurs qui utilisent de la cryptologie. Ces applications utilisent des algorithmes de chiffrement (comme RSA par exemple), utilisant un système de 2 clés (une dite *privée* et l'autre *publique*) qui permettent de chiffrer les données sur le réseau. En bref, ces algorithmes à clefs asymétriques reposent sur des propriétés mathématiques de factorisation des grandes nombres premiers. Ils permettent de faire en sorte que tout ce qui est chiffré avec la clé privée, peut être déchiffré avec la clef publique correspondante, et réciproquement. Ce chiffrement permet de rendre illisibles les données qui sont échangées entre un client et un serveur, et difficilement déchiffrables par un tiers qui n'aurait pas la clé correspondante. En outre ce système de chiffrage par des "bi-clés" permet d'apporter des fonctionnalités d'authentification pour des serveurs ou des individus. En effet si je peux déchiffrer un message avec la clé publique de X, c'est que seul X a pu l'écrire avec sa clé privée.

Principales applications utilisant la cryptologie

Par défaut, les échanges qui se font sur l'internet entre un programme client et une application serveur se font "en clair", laissant ainsi toute possibilité à un individu malveillant d'intercepter et lire ce qui est échangé lors d'une connexion avec un serveur distant (www ou mail ou ftp...). Ainsi lorsque vous tapez votre mot de passe sur votre poste, en vue de relever votre courrier électronique, ou bien d'accéder à un site bancaire via le Web, une possibilité existe que votre mot de passe soit lu par une tierce partie. On imagine alors les conséquences néfastes sur la sécurité des systèmes informatiques, et votre vie privée. Cette possibilité est désormais inacceptable, et il convient de la rendre nulle en utilisant des programmes utilisant du chiffrement.

Les programmes utilisant des algorithmes de chiffrement basés sur des clés existent depuis plusieurs années, mais leur utilisation avec des clés de 128 bits en France était jusqu'à il y a peu de temps, légalement prohibée. Ce n'est que récemment (Février 2002) que le CNRS a obtenu pour 5 ans l'autorisation légale d'utiliser certains moyens de chiffrement avec des clés de 128bits (logiciel ssh).

Il est désormais nécessaire de privilégier et utiliser toute application permettant un chiffrage du dialogue

client-serveur. Dans ce système, en bref, les parties "cliente" et "serveur" chiffrent, les données (resp. déchiffrent) par un système de clé privée/clé publique. Pour l'utilisateur final, ce procédé ne nécessite aucune action supplémentaire ou complexe. Il est juste nécessaire d'utiliser des applications clientes susceptibles d'engager une transaction chiffrée.. La majeure partie du travail se fait du coté du serveur (et donc pour l'administrateur du serveur).

Ce mode de transmission s'appelle **SSL** (Secure Socket Layer), normalisé en **TLS** (Transport Layer System). Que peut-on sécuriser avec les applications utilisant SSL/TLS ?

– le relevé de son courrier électronique : Seuls les programmes de mail suivants proposent un dialogue sécurisé par SSL/TLS. Il s'agit de "Outlook Express" ou "TheBat" sur Windows, "Netscape messenger" (toute plateforme), Kmail ou Mutt (sur Linux). En ce qui concerne Eudora, seule les versions supérieures à 5.1 proposent ce chiffrement. Pour utiliser ce chiffrement sur vos programmes, il vous suffit de chercher dans les options de configuration du programme, la case indiquant "utiliser une connexion sure SSL". Bien entendu il faudra que du coté du serveur de messagerie, l'administrateur du système ait installé les serveurs d'accès aux boites de Mail, permettant le chiffrage (POPS et/ou IMAPS).

– L'accès à un site web : Pour sécuriser les échanges avec un serveur web distant, il suffit juste pour le client d'utiliser la syntaxe https en lieu et place de http lors de la connexion à un site web. Ainsi `https://www.camif.fr/` pemettra d'accéder au server WWW d'un site en mode sécurisé, les échanges étant chiffrés par le serveur. Du coté du serveur, il est bien entendu nécessaire que l'administrateur du site distant ait mis en place un serveur WWW sécurisé supportant SSL, sans quoi la connexion sera refusée.

– La connexion sur une machine distante : il est question là, de se connecter et travailler sur une machine B, alors que l'on se trouve devant une autre machine A distante. On peut donc utiliser au maximum les potentialités d'un réseau de machines. Depuis longtemps Linux (et tous les Unix) permettent de travailler en réseau en utilisant les processeurs et disques de machines différentes. Il y a quelques temps, les programmes de base s'appelaient *telnet* et *rlogin*. Ces programmes souffrent d'un certain nombre de faiblesses en partie dues au fait que les échanges, encore une fois, entre client et serveur sont potentiellement lisibles avec un analyseur de réseau. Il est désormais indispensable de reléguer aux oubliettes l'utilisation de ces 2 anciens programmes de connexion. D'autant qu'un substitut parfait est disponible ! **SSH** pour Secure Shell (ou SSF pour sa version française utilisant des clés inférieures à 128bits)

– L'échange de fichiers par ftp : là encore le dialogue client-serveur FTP peut être chiffré par *sftp* grâce aux serveurs SSHv2. Certains applications clientes (www.ssh.org) offrent des accès et transferts très ergonomiques.

Contrôler l'intégrité de son système (`tripwire, rpm -V`)

En cas d'intrusion sur votre système Linux, il est fréquent que les pirates installent ce que l'on appelle un *rootkit* (disponibles sur Internet). Un *rootkit* (boite à outils d'un faux administrateur malveillant) est un ensemble de logiciels qui vont permettre à un intrus de s'introduire sur un système cible, et le modifier en laissant le moins de traces possibles. Le but pour l'intrus étant d'agir, en passant inaperçu du vrai administrateur. Par exemple, la commande `ls` sera modifiée pour ne pas laisser apparaître certains fichiers du rootkit lui même. La commande `ps` cachera certains processus, la commande `netstat` n'affichera pas certaines connexions ouvertes, etc. Le *cracker* peut alors facilement installer un accès caché au système (encore appelé *backdoor*), sans que l'administrateur puisse le déceler. Dans ce contexte il très est difficile de déceler une intrusion par les moyens de base. Il est donc indispensable d'utiliser des systèmes de contrôle d'intégrité qui permettent de savoir quels fichiers du système ont pu être modifiés.

Le paquetage **tripwire** (http://www.tripwire.org) est un système d'utilisation assez simple, qui permet de vérifier l'intégrité d'un système Linux. Il permet de contrôler toute modification du système par rapport à un état de base sain. Tous les fichiers rajoutés, détruits, ou modifiés seront décelés. tripwire est donc un programme de détection d'intrusion dans la mesure ou il permet de vérifier très rapidement l'état d'un système après intrusion, et repérer si un intrus a modifié le système.

Apres avoir initialisé le système par la commande `/etc/tripwire/twinstall.sh` Le principe de base consiste :
– à calculer une image de base (encore appelée *empreinte* ou *condensé*) du système lors de l'installation originelle. Cette empreinte est chiffrée et conservée dans un fichier.
– recalculer l'empreinte du système régulièrement (chaque jour) et la comparer à l'empreinte originale. tripwire indique alors toutes les modifications entre les 2 états !

tripwire est assez simple à utiliser :

– `/etc/tripwire/twinstall.sh` : étape préliminaire consistant à générer les clés de chiffrage qui vont permettre de chiffrer les fichiers de rapport.
– `tripwire --init` : calcul de l'empreinte initiale (condensé) du système. Cette empreinte est stockée dans un fichier chiffré, appelé
`/var/lib/tripwire/<votremachine.domaine>.twd`
– `/usr/sbin/tripwire --check` : lance un contrôle d'intégrité. La base de données initiale sert de point de comparaison. Le diagnostic donne ce qui a été modifié entre les 2 contrôles. La commande /usr/sbin/tripwire â€"check produit un fichier journalier de rapport chiffré qui se trouve dans `/var/lib/tripwire/report`

Il est nécessaire de lancer régulièrement les contrôles d'intégrités. tripwire peut être lancé régulièrement par l'ordonnanceur de tâches (**cron**) de Linux. Un rapport vous est envoyé par mail.

Une autre manière permettant de vérifier l'intégrité d'un ensemble de fichiers, est de s'appuyer sur les gestionnaires de paquetages `rpm`. L'option -V de la commande rpm indique si le paquetage installé a été corrompu par rapport à la version de base.

`rpm -Va` ou `rpm -V <nom-dupaquetage>` compare les informations des fichiers installés sur le disque avec les informations prises dans la base de données des rpm. Le contrôle vérifie la taille des fichiers, le checksum, les permissions, les propriétaires de chaque fichier du paquetage. Exemple :

```
#rpm -qf  /bin/ls    fileutils-4.0-13mdk  # rpm -V fileutils  # touch /bin/ls
```

(on modifie la commande ls)

```
# rpm -V fileutils
```

(la vérification par rpm indique cette modificaiton)

```
.......T /bin/ls
```

(#T indique un changement dans la date de modification de la commande..)

Mettre à jour rapidement une application avec Linux ?

En cas de réception d'un avis de sécurité (l'administrateur système d'un laboratoire CNRS devrait logiquement en être informé par le biais des CERT-Renater ou CERT-A) informant d'une faille critique sur un logiciel. Il est alors nécessaire de faire rapidement une mise à jour du paquetage incriminé. Les corrections dans la communauté du logiciel libre étant assez rapide à être faites, il faut alors se procurer la version corrigée sur certains sites officiels de l'Internet (debian.org, mandrake.com, rpmfind.net, redhat.com, isc.org etc..) et l'installer. Cela dépend de la distribution que vous utilisez :

– par exemple, *DrakConf* (sur une distribution Mandrake), choix **mise à jour des logiciels**, ou plus directement MandrakeUpdate automatise totalement la récupération et l'installation d'une version corrigée d'un logiciel.
– De manière moins directe, mais tout aussi efficace, le site http://www.rpmfind.net permet de chercher, et récupérer la dernière version d'un programme corrigé. Une fois le paquetage récupéré, il suffit de lancer la mettre à jour sur votre système, par la commande rpm $`rpm -Uvh <paquetage-version.rpm>`

Voici quelques autres commandes bien utiles pour bien gérer vos paquetages installés avec le système de commandes RPM

```
$ rpm -qa
```

lister tous les paquetages installés sur mon PC

```
$ rpm -e <nom-du-paquetage>
```

enlever/désinstaller un paquetage

```
$ rpm -qil <nom-du-paquetage>
```

lister le contenu d'un paquetage déjà installé

```
$ rpm -qilp <nom-paquetage-version.rpm>
```

lister le contenu d'un paquetage non encore installé (encore sous forme de fichier .rpm)

```
$ rpm -qf /chemin/vers/le/fichier
```

savoir à quel paquetage appartient un certain fichier ?

```
$ rpm -ivh <paquetage-version.rpm>
```

Installer un paquetage

```
$ rpm -Uvh <paquetage-version.rpm>
```

mettre à jour un paquetage

```
$ rpm -Va ou rpm -V
```

vérification de l'intégrité du paquetage

Un doute sur la fonction du paquetage inn ?

– `$ rpm -qi inn` vous indiquera à quoi sert le paquetage
– `$ rpm -qil inn` : indique la composition du paquetage (s'il est déjà installé sur la machine)
– `$ rpm -qilp /mnt/cdrom/Mandrake/RPMS/inn-2.2.2-8mdk.i586.rpm` indique la composition du paquettage rpm inn, celui ci n'étant pas encore installé sur la machine

rajouter après coup le paquetage tcpdump que j'ai oublié lors de l'installation ?

```
$ mount /mnt/cdrom $ rpm -ivh /mnt/cdrom/Mandrake/RPMS/tcpdump-3.6.2-2mdk.i586.rpm
```

Sécurité Linux avancée : filtrage au niveau du transport réseau (ipchains)

Enfin, dernier point qui mérite d'être signalé, le système Linux a depuis longtemps la possibilité de faire du filtrage de trames réseau TCP/IP. Cela signifie que, Linux a la capacité d'accepter ou de refuser de traiter des trames réseau qui lui proviennent, en fonction de plusieurs critères : protocole réseau utilisé, adresses sources, du port applicatif qui est demandé. Ce système de filtrage au niveau du noyau Linux s'appelle **ipchains** pour la série des noyaux Linux **2.2.***, ou **iptables** pour la série des noyaux **2.4.***.

On peut donc faire du filtrage d'accès au niveau des demandes de connexion réseau provenant de l'extérieur. Cela est un peu plus général et performant que le filtrage au niveau des applications que nous avons cité plus haut (par tcpwrapper et xinetd). En effet, à ce niveau on peut intervenir directement au niveau des couches de transport réseau et des protocoles comme TCP, UDP, ICMP, mais également au niveau de n'importe quelle application. On peut en outre différencier des connexions entrantes (input), sortantes (output), ou transmises (forward) dans le cas ou le PC Linux possède 2 interfaces réseau et agit comme un routeur.

Ce système permet également de faire la différence entre des demandes de connexions issues de l'intérieur du site, et celles issues de l'extérieur (connexion dites "established"). Ce système est un peu plus complexe à mettre en oeuvre, et demande une bonne connaissance du protocole TCP/IP, et des ports applicatifs... mais il est très efficace.

– **par défaut**, le noyau Linux **accepte** tout en entrée, sortie, et redirection.

```
$ /sbin/ipchains -L  Chain input (policy ACCEPT):  Chain forward (policy ACCEPT):  Chain output (policy ACCEPT):
```

– Voici quelques options principales pour écrire une chaîne de filtre :

```
$ /sbin/ipchains   -A input [rajoute une chaîne de filtre en entrée] -s adresse-source 0/0 [contrôle sur l'adresse source en notation CIDR] -d adresse-destination [contrôle sur l'adresse destination en notation CIDR] numéro-de-port protocole  -y [indique que la demande connexion est initiée depuis l'extérieur] -j ACCEPT|DENY|REJECT [politique de filtrage à mettre en place] -l [pour tracer la règle de filtrage dans un fichier de journalisation]
```

– exemple d'un fichier de configuration de ipchains dans

```
/etc/sysconfig/ipchains.   ## accepte toute connexion entrante de l'exétérieur sur le port 22 = ssh -A input -p tcp -s 0/0 -d 0/0 22 -y -j ACCEPT   # autoriser tout tcp et udp depuis le site de confiance 138.125.2 entre les ports 0 et 1023 -A input -p tcp -s 138.125.2.0/24 -d 0/0 0:1023 -y -j ACCEPT  -A input -p tcp -s 139.124.16.0/24 -d 0/0 0:1023 -y -j ACCEPT  -A input -p udp -s 139.124.2.0/24 -d 0/0 0:1023 -j ACCEPT  -A input -p udp -s 139.124.16.0/24 -d 0/0 0:1023 -j ACCEPT   # rejeter tout le reste  -A input -p tcp -s 0/0 -d 0/0 0:1023 -y -j REJECT -l  -A input -p tcp -s 0/0 -d 0/0 2049 -y -j REJECT -l  -A input -p udp -s 0/0 -d 0/0 0:1023 -j REJECT -l  -A input -p udp -s 0/0 -d 0/0 2049 -j REJECT -l   ## rejette X11 et xfs -A input -p tcp -s 0/0 -d 0/0 6000:6009 -y -j REJECT -l  -A input -p tcp -s 0/0 -d 0/0 7100 -y -j REJECT -l
```

Grâce à ce type de filtre opéré par le noyau Linux, on voit par un logiciel de scan comme *nmap*, qu'un certain nombre d'applications sont inaccessibles ("filtered") depuis une machine de l'extérieur. Avec l'option DENY à la place

de REJECT, la mesure est plus sévère puisque aucune indication de rejet n'est fournie. L'indication "filtered" n'apparait pas, la machine protégée devient alors un "trou noir" qui ne laisse apparaître aucun des services qu'elle met en oeuvre.

```
$nmap sonpc.sonreseau.fr  Interesting ports on monpc.monreseau.fr (138.125.1.101):  (The 1511 ports scanned but not shown below are in state: closed)  Port State Service  21/tcp filtered ftp  22/tcp open ssh  25/tcp filtered smtp  67/tcp filtered bootps  68/tcp filtered bootpc  69/tcp filtered tftp  80/tcp filtered http  98/tcp filtered linuxconf  110/tcp filtered pop-3  111/tcp open sunrpc  123/tcp filtered ntp
```

Conclusions

Ces quelques conseils sont relatifs à la sécurité de base sur des machines de type unix. Ils vous permettront de rester connecter au réseau mondial avec une bonne sécurité de base, et de tirer tous les bénéfices et avantages de Linux et des "logiciels libres" sans trop de dommages...

Mais bien entendu ces mesures ne sont pas suffisantes à elles seules, pour bâtir une politique de sécurité totale et cohérente sur l'ensemble d'un laboratoire ou d'un campus. Il est nécessaire de consolider les quelques mesures que nous avons passé en revue, par la mise en oeuvre d'autres mesures relatives à l'architecture physique des réseaux, à des filtrages en entrée de site, jusqu'à certaines mesures relevant du comportement individuel (chartes informatiques), mais cela dépasse le cadre de cet article.

Quelques références

– http://www.urec.fr/securite/
– http://www.cru.fr/securite/
– http://www.linuxsecurity.com
– http://www.xinetd.org
– http://www.fr.linuxfocus.org/Francais/November2000/article175.shtml
– http://www.tripwire.org
– http://perso.univ-rennes1.fr/bernard.perrot/SSF/
– http://www.openssh.org
– http://www.ssh.com

Post-scriptum :

L'article original de Maurice Libes du Centre d'Océanologie de Marseille est disponible au format RTF .