TUTO SECURITE: LES BASES.

LES DROITS D'ACCÈS

Dans l'environnement du système d'exploitation Unix, les fichiers et les répertoires sont organisés dans une structure arborescente avec des modes d'accès spécifiques. Le réglage de ces modes, par le jeu des bits de droits d'accès (en tant que chiffres octaux), est à la base de la sécurité du système Unix. Les bits de droits d'accès déterminent comment les utilisateurs peuvent accéder aux fichiers et le type d'accès auquel ils sont autorisés. Il existe trois types de droits d'accès utilisateur pour tous les fichiers et les répertoires sous Unix : le propriétaire, le groupe et les autres. Les droits d'accès à l'écriture, la lecture et l'exécution pour chacun des types d'utilisateur sont aussi contrôlés par les bits de droits d'accès (Schéma 1). La flexibilité en matière de sécurité des fichiers est avantageuse, mais elle a été critiquée comme vecteur de failles dans la sécurité du système.

Puisque le mécanisme de protection des fichiers est si important dans le système d'exploitation Unix, il est évident qu'un réglage approprié des bits de droits d'accès est requis pour une sécurité globale. Hormis l'ignorance de l'utilisateur, les cas de compromission de fichier les plus communs sont dus au réglage par défaut des bits de droit d'accès à la création des fichiers. Pour certains systèmes, ce réglage par défaut est l'octal 644, ce qui signifie que seul le propriétaire du fichier peut y écrire et y lire, pendant que tous les autres ne peuvent qu'y accéder en lecture. (3) Dans nombre d'environnements « ouverts », ceci peut être acceptable. Cependant, au cas ou il existe des données sensibles, l'accès en lecture pour les autres devrait être désactivé. La commande umask répond en fait à cette exigence. Un réglage conseillé d'umask à 027 activerait tous les droits d'accès pour le propriétaire du fichier, ôterait le droit d'écriture au groupe et les tous les droits pour les autres (octal 750). En insérant cette commande umask dans le fichier .profile ou .login d'un utilisateur, le réglage par défaut sera écrasé par le nouveau paramétrage à la création d'un fichier.

SUID, SGID ET STICKY BIT

Les fichiers contenus dans un répertoire dont le sticky bit est positionné, peuvent être renommés ou effacés uniquement par leur propriétaire, le propriétaire du répertoire ou le super-utilisateur.

Le set user id (suid) et le set group id (sgid) identifient les droits d'accès d'un fichier pour l'utilisateur et le groupe. En plaçant les bits de droits d'accès suid ou sgid sur un fichier exécutable, d'autres utilisateurs peuvent obtenir un accès aux mêmes ressources (par l'intermédiaire du fichier exécutable) comme le réel propriétaire du fichier.

Par exemple :

Supposons que le programme bob.x de Bob soit un fichier exécutable accessible aux autres. Quand Mary exécute bob.x, Mary devient la nouvelle propriétaire du programme. Si pendant l'exécution du programme, bob.x demande à accéder au fichier browse.txt, alors Mary doit avoir des droits d'accès en lecture ou en écriture antérieurs sur browse.txt. Cela donnerait à Mary et à tous les autres un accès total sur le contenu de browse.txt, même si elle n'exécute pas bob.x. En activant le bit suid sur bob.x, Mary aura les mêmes autorisations d'accès à browse.txt que le réel propriétaire du programme, mais elle aura seulement accès à browse.txt pendant l'exécution de bob.x. Par conséquent, en ajoutant suid ou sgid, on évitera d'afficher malencontreusement des fichiers tels que browse.txt.

Bien que ce dispositif semble offrir un contrôle d'accès substantiel au système de fichiers sous Unix, il n'est pas exempt d'un DEFAUT MAJEUR. Il est toujours possible que le super-utilisateur (l'administrateur du système) puisse avoir un dossier accessible en écriture pour les autres pour lequel est aussi activé le suid. En modifiant le code du programme (un hacker par exemple), un tel fichier exécutable permettrait à un utilisateur de devenir administrateur. Très prochainement, cet intrus pourrait compromettre complètement la sécurité du système et le rendre inaccessible, y compris aux autres administrateurs. Ainsi que Farrow (5) le souligne, « (...) avoir une copie set user id d'un shell dont le propriétaire est root est meilleur que de connaître le mot de passe root ». Pour corriger cette faille de sécurité, les fichiers suid accessibles en écriture devraient être recherchés et supprimés par l'administrateur du système. La détection de tels fichiers par les utilisateurs normaux est aussi essentielle pour corriger des brèches de sécurité existantes.

--> set-UID et set-GID sont à éviter

MOTS DE PASSE

L'authentification des utilisateurs sous Unix s'exerce grâce aux mots de passe personnels. Bien que des mots de passe ajoutent un niveau de sécurité supplémentaire aux contraintes physiques, ils sont eux-mêmes facteurs de la plupart des compromissions d'un système informatique. Le manque de conscience et de responsabilité des utilisateurs contribue grandement à cette forme d'insécurité informatique. Cela est vrai pour de nombreuses installations informatiques, ou l'identification par mot de passe, l'authentification et l'autorisation sont exigés pour accéder aux ressources—et le système d'exploitation Unix ne fait pas exception.

Les informations de mots de passe dans nombre de systèmes à temps partagé sont placées dans des fichiers à usage restreint et qui ne sont pas habituellement lisibles par les utilisateurs. Le système Unix diffère à cet égard, puisqu'il permet à tous les utilisateurs d'avoir un accès en lecture au fichier /etc/passwd (Schéma 2) ou sont stockés des mots de passe chiffrés et d'autres informations sur les utilisateurs. Bien que le système Unix applique une méthode de chiffrage dite à sens unique, et dans la plupart des systèmes une version de norme de chiffrement de données modifiée (DES, data encryption standard, norme de chiffrement de données), on connaît des méthodes pour casser des mots de passe. Parmi ces méthodes, les attaques par brute-force sont généralement les moins efficaces, tandis que les techniques liées à l'heuristique (des

conjectures et des connaissances satisfaisantes en matière de mots de passe) ont tendance à réussir. Par exemple, le fichier /etc/passwd contient des informations utiles telles que le nom d'utilisateur et des champs de commentaire. Les noms d'utilisateur sont particulièrement fructueux pour les « casseurs de mot de passe », puisque nombre d'utilisateurs emploient des variantes de ces noms pour leur mots de passe (orthographe inverse, ajout d'un seul chiffre, etc.).

Une * en lieu et place du mot de passe crypté signifie sur les systèmes traditionnels que cet utilisateur a un compte verrouillé. C'est le cas de bin par exemple.

La première mesure à prendre pour protéger les mots de passe d'un système UNIX est d'adopter un mécanisme appelé shadow password.

Avec ce principe, les chaînes cryptées contenant les mots de passe d'utilisateurs et de groupes ne sont plus lisibles par tout un chacun dans les fichiers /etc/passwd et /etc/group. Elles sont stockées ailleurs, consultables et modifiables seulement par un certain nombre d'utilisateurs (dont root).

Un éventuel pirate, ayant un compte sur votre machine et voulant y essayer son dictionnaire, sera alors obligé de s'affronter à l'invite du login. Les nombreux messages d'avertissements enregistrés par le système, dont un exemple figure ci-dessous, ne manqueront pas dans ce cas de vous alerter.

La variable d'environnement PATH

PATH est une variable d'environnement qui pointe sur une liste de répertoires, qui est recherchée quand un fichier est requis par un processus. L'ordre de cette recherche est indiquée par la séquence des répertoires listés dans le nom du PATH. Cette variable est établie lors de l'ouverture de session de l'utilisateur et elle est définie dans les fichiers .bash_profile ou .bashrc à la racine du répertoire de l'utilisateur.

Si un utilisateur place le répertoire courant en première place dans le PATH, alors les programmes du répertoire courant seront exécutés en premier. Des programmes homonymes dans d'autres répertoires seront ignorés. Bien que l'accès aux fichiers et aux répertoires soient plus aisés avec une variable PATH définie de cette manière, il peut exposer un utilisateur à des chevaux de Troie préexistants.

Pour illustrer ceci, supposons qu'un cheval de Troie, semblable au programme cat, contienne une instruction qui donne des droits d'accès privilégiés à un assaillant. Le programme cat factice est placé dans un répertoire public /usr/his , dans lequel un utilisateur travaille souvent. À présent, si l'utilisateur dispose d'une variable Path avec en premier le répertoire courant, et qu'il exécute la commande cat dans /usr/his, le programme cat factice dans /usr/his sera exécuté mais non la commande système cat située dans /bin.

Afin d'éviter ce genre de type de violation du système, la variable Path doit être correctement définie. D'abord, si cela est possible, excluez le répertoire courant en tant que premier élément dans la variable Path et saisissez le nom complet du chemin quand vous invoquez des commandes système Unix. Ceci accroît la sécurité des fichiers, même s'il est moins pratique de

travailler avec . Ensuite, si le répertoire courant doit être inclus dans la variable Path, alors il devrait toujours se trouver en dernier. De cette manière, des programmes tels que vi, cat, su et ls seront exécutés en premier depuis les répertoires systèmes tels que /bin et /usr/bin avant de chercher dans le répertoire courant de l'utilisateur.

Le cas du compte root

Root, c'est le super utilisateur par excellence. Sous Unix, il peut tout faire, lemeuilleur comme le pire. Il a les pleins pouvoirs, et comme personne n'est au dessus de root, personne ne peut limiter l'étendue des dégats qu'il s'apprête à commettre.

Mais comment peut-on en arriver à faire des dégats ?

1°)

Le premier cas, est le cas des commandes fatales. Celles dont on ne revient pas. Par exemple, un rm *, ou pire un rm-r *. Le premier est une suppression simple de tout fichier du répertoire courant. Le deuxième est fait la même chose, sauf que l'option -r permet de faire un récursivité, et donc déscendra dans les sous-répertoires et supprimera les fichiers.

En root, il existe une sécurité, en root la commande rm est un alias qui qui force la vrai commande rm à faire une demande de confirmation avant d'effectuer la supression réel du répertoire. Malgré tout, si notre commande rm -r * devient rm -rf *, il ne demandera plus de confirmation (-f = force). Pour ceux qui doutent de la probabilité de pouvoir jamais taper cette commande, qu'ils considèrent le cas de quelqu'un qui souhaite retirer tout les fichiers HTML, via un "rm -rf *.mp3", pour faire un peu de place, et que par inadvertance, il tape un espace de trop, entre l'étoile et le point (rm - rf * .mp3); comme par hasard, cela lui ferait effacer tous les fichiers, plus le répertoire courant. Autre méthode pour se tirer dans le pied, une variante dont je vous laisse deviner les effets : "rm -rf /home/users/toto/pile.* ". Imaginez que vous lanciez cette commande dans "/"...

2°)

Le programme que nous avons télécharger se décompresse bien dans un répertoire à part, pas de problème de compilation, il n'y à plus qu'à lancer. Vous avez alors plusieurs cas possibles. Dans le cas le plus fréquent (et heureusement le plus fréquent) il ne se passe rien de notable. L'application fait son travail et rien d'autres. Les autres cas marginaux, devraient tout de même vous faire réfléchir. Et si, en s'installant, cette nouvelle configuration "écraisait" votre configuration actuelle? Et si un utilisateur avait modifié le code pour y introduire un cheval de troie ou autre virus ... Pardon ? Vous relisez tout le code avant de l'éxécuter, c'est bien, vous êtes donc à l'abri de ce genre de de soucis. Mais, si vous êtes "normal" (end-user dans le jargon), vous venez de laisser un grand trou dans vos défenses, car root à le droit de tout faire sur votre machine, y compris celui de modifier des droits sur des fichiers, changer le contenu d'autres

fichier, remplacer des fichiers par d'autres, mettre en place un service "pirate" sur n'importe quel port, et eventuellement écouter ce qui se passe sur votre réseau.

3°)

Regardons le problème sous un autre angle : généralement, si vous êtes root, c'est que c'est quand même plus pratique. Donc, toujpours dans le même esprit, vous avez tendance à chanegr les droits système pour que cela veuille bien "tourner". Ceux qui manipulent des modems, vont modifier des droits dans /dev. Ou encore vont donner des droits "root", via l'utilisation sauvage du bit "s", ou du "sticky bit" (bit collant), à des programmes qui auraient fort bien pu s'en passer. Pour information, le bit "s" est un droit que l'on donne à un exécutable, pour qu'il qu'il se lance non pas avec les droits de la personne qui l'invoque, ou du groupe qui le possède. C'est notamment grâce à ce mécanisme qu'un utilisateur normal peut changer son mot de passe. Je pense ici encore à des exécutables de télécom, comme xeux qui gèrent le modem. On a vu certains changer les droits sur le démon sendmail, par exemple afin de lui permettre de s'exécuter correctement, ou bien de donenr ce genre de privilège à des daemons systèmes, lesquels lorsuq'ils plantent sont susceptibles d'ouvrir des failles béantes dans votre systèmes.

Partage de fichiers via NFS

NFS fonctionne bien pour partager des systèmes de fichiers entiers avec de nombreux hôtes connus et de façon très transparente. Nombre d'utilisateurs ont accès à des fichiers sur un montage NFS sans même s'apercevoir que le système de fichiers qu'ils utilisent n'appartient pas à leur système local. Cependant, la facilité d'utilisation va de paire avec toute une série de problèmes potentiels de sécurité.

Accès hôte :

NFS contrôle qui peut monter et exporter un système de fichiers en fonction de l'hôte présentant la demande de montage et non en fonction de l'utilisateur qui utilise le système de fichiers. Les hôtes doivent obtenir des droits explicites pour monter un système de fichiers. Le contrôle d'accès n'est cependant pas possible pour les utilisateurs, si ce n'est les autorisations de fichier et de répertoire. Autrement dit, lorsque vous exportez un système de fichiers via NFS vers un hôte distant, vous ne faites pas confiance uniquement à l'hôte qui monte le système de fichiers, mais aussi aux utilisateurs qui ont accès à cet hôte et, par conséquent, à votre système de fichiers. Ce risque peut être contrôlé, en exigeant des montages en lecture seule par exemple ou en réduisant (squash) les utilisateurs à un ID utilisateur et groupe commun, ce qui peut empêcher toutefois que le montage ne soit utilisé de la façon souhaitée à l'origine.

De plus, si l'auteur d'une attaque prend le contrôle du serveur DNS utilisé par le système qui exporte le système de fichiers NFS, le système associé à un nom d'hôte spécifique ou à un nom de domaine pleinement qualifié peut être pointé vers un ordinateur non autorisé. A ce stade, l'ordinateur non autorisé est le système autorisé à monter le partage NFS car aucune information de nom d'utilisateur ou de mot de passe n'est échangée pour fournir plus de sécurité au montage NFS. Les mêmes risques s'appliquent aux serveurs NIS compromis, si les groupes réseau NIS sont

utilisés pour permettre à certains hôtes de monter un partage NFS. En utilisant une adresse IP dans /etc/exports, ce genre d'attaque est plus difficile.

Les caractères spéciaux devraient être utilisés avec modération lorsque vous accordez l'accès à un hôte à un partage NFS. La portée du caractère spécial peut englober des systèmes dont vous ignorez l'existence et qui ne devraient pas être autorisés à monter le système de fichiers.

Autorisations de fichier :

Lorsqu'un système de fichiers est monté en lecture-écriture par un hôte distant, la seule protection pour les fichiers partagés repose dans leurs autorisations et leur appartenance à un utilisateur et un groupe. Si deux utilisateurs partageant la même valeur d'ID utilisateur montent le même système de fichiers NFS, ils pourront modifier leurs fichiers l'un et l'autre. De plus, tout individu connecté en tant que super-utilisateur sur le système client peut utiliser la commande su pour devenir un utilisateur ayant accès à des fichiers spécifiques via le partage NFS.

Le comportement par défaut lors de l'exportation d'un système de fichiers via NFS est d'utiliser root squashing. Cela a pour effet de paramétrer sur une valeur du compte nobody du serveur l'ID utilisateur de tout utilisateur se servant du partage NFS en tant que super-utilisateur sur son ordinateur local . Vous ne devriez pas désactiver la réduction du super-utilisateur, à moins que la présence de nombreux super-utilisateurs sur votre système ne vous dérange pas.

Si vous n'autorisez les utilisateurs qu'à lire les fichiers via votre partage NFS, vous pourriez utiliser l'option all_squash, qui fait en sorte que tous les utilisateurs accédant à votre système de fichiers exporté aient l'ID utilisateur de l'utilisateur nobody.

TUTO SECURITE : VULNERABILITES DES APPLICATIONS.

Certains logiciels installent sur votre ordinateur un serveur HTTP permettant d'afficher des pages Web. C'est notamment le cas de quelques, éditeurs de pages Web tel que Microsoft Front Page. Outre le fait qu'il soit strictement interdit, par contrat Urbanet de diffuser des données de tout type de serveur vers le réseau Internet, l'installation de tels serveurs sur votre système ouvre souvent de nombreuses portes aux intrusions. C'est notamment le cas de nombreux programmes CGI installés implicitement sur les serveurs Web ou développés ultérieurement par des programmeurs.

D'autres types de serveurs ou de services ouvrent aussi des portes d'entrée dans votre système. À moins d'être familier avec leur configuration, de télécharger les plus récentes mises à jour et de se tenir régulièrement informé des dernières brèches de sécurité qui sont périodiquement

découvertes, l'installation de serveurs Proxy, FTP (en particulier FTP anonyme), SMTP, Sendmail, les services à distance (rsh, rlogin, etc.) et autres du même genre offrent de grands risques de créer des failles dans votre système. N'activez pas de tels services pour un ordinateur personnel

FTP ANONYME

Utilisé pour éviter les dangers de l'upload (virus,troyens,scripts). L'utilisateur peut se connecter via un compte anonyme :

login = anonymous

pwd = anonymous ou l'adresse mail

L'idéal est de créer un répertoire pour le download en anonymous, ce répertoire doit avoir comme propriétaire root pour eviter

l'upload de fichiers.

NB : cp-p : copier le fichier de qq d'autre en gardant ses propres permissions.

Si il est quand même nécessaire d'autoriser l'upload, malgré que cela ne soit pas conseillé, il faut utiliser un filsesystem différent

Si vous voulez que les utilisateurs anonymes puissent déposer des fichiers, créez le répertoire ~ftp/pub/incoming (propriétaire root, droits 733). Faîtes un ``chmod +t ~ftp/pub/incoming ''. Normalement le démon FTP interdit aux utilisateurs anonymes d'écraser un fichier existant, mais un utilisateur normal pourrait détruire n'importe quoi. En mettant les droits à 1733 ce ne sera plus possible. Avec wuftpd vous pouvez configurer le démon pour que les fichiers créés le soient avec les droits 600 et appartiennent à root (ou tout autre utilisateur).

Parfois répertoires ``incoming'' sont utilisés frauduleusement pour échanger de fichiers piratés ou pornographiques. Les fraudeurs y créent des sous-répertoires cachés précisément dans ce but. Ça aide un peu de rendre le répertoire incoming illisible par les utilisateurs anonymes. Avec les serveurs FTP usuels, on ne peut pas empêcher la création de répertoires dans incoming. En fixant les droits qu'on veut à ces sous-répertoires on peut ainsi partager des fichiers visibles par n'importe qui.

DÉMONS SERVEURS

Mettre les privilèges minmums: ne pas faire tourner les serveurs en root, car sinon via un naviguateur on peut faire passer une commande.

Attaques symlink

Ce type d'attaques consiste a crééer dans /tmp un lien symbolique d'un fichier temporaire qu'une application va peut-être créer. Ce lien symbolique pointe vers un fichier arbitraire (par ex : pourquoie pas /etc/passwd?). Lorsque l'appli créera son propre lien, si cette appli est lancée par root, elle écrasera le lien existant et par la le fichier.

Pfs passer à une nouvelle version de l'application ne suffit pas toujours, le makefile doit etre modifie car la configuration par défaut est dangereuse (ex : programme "majodormo")

Sendmail (service de mail de l'internet)

Pourquoi Sendmail utilise classiquement root ?

Plusieurs raisons essentielles permettent de jusitifier l'exécution de Sendmail en tant que root, notamment :

- Sendmail écoute sur le port 25, port privilégié (inférieur à 1024).

- Les utilisateurs génèrent des fichiers .forward dans leur répertoire personnel afin de transférer leurs messages vers d'autres adresses SMTP. Sendmail doit pouvoir accéder à ce fichier afin d'en lire le contenu, même si les utilisateurs ont disposé un droit d'accès restrictif sur leur répertoire personnel (mode 700).

- Sendmail doit pouvoir devenir temporairement propriétaire des fichiers de listes de messagerie (mailing-lists) gérées par les utilisateurs (en prenant l'identité de l'utilisateur).

Le fichier .forward situé dans le répertoire courant d'un utilisateur d'un serveur de messagerie sendmail, permet de mettre en oeuvre un mécanisme de redirection des messages.

*La variable "URFY" vérifie si l'utilisateur existe bien, est bien créé dans la DB du mailer --> pas désactiver

*Attention à la longeur maximale des messages --> perte d'efficacité des serveurs

*Relais SMTP : il faut bien configuere le serveur mail pour éviter qu'il relaire du spam par ex. Une connexion VPN peut passer

outre cette protection --> rejeter les relais SMTP non désirés

*Lors d'une réinstallation, il faut recréer les liens symboliques pour être sûr que c'est la nouvelle version qui sera utilisée

[Différentes versions de sendmail]

-> 8.9.3 : exécuté sous l'identité d'un groupe "system" et le programme n'est plus que SetGID

-> 8.12 : groupe dédié "smmsp". Intérête de l'open source : si une faille est découverte, il est très rapidement mis à jour. Le répertoire /var/spool/clientmqueue est ajouté pour les clients (droit d'écriture pour le groupe "smmsp" et c'est le processus sendmail qui déplace vers la vraie file d'attente var/spool/mqueue)

*Il existe une option pour que sendmail refuse de s'exécuter avce des droits sur les répertoires trop légers --> option DontBlameSendmail

*Si sendmail a besoina bsolument de droits d'accès sur des répertoires, les laisser tel quel et utiliser d'autres répertoires pour les fichiers de config

*fichiers .mc :

- utiliser un shell restreint
- appliquer des filtres
- définir des alias pour les répertoires

*MIMEDefang : filtre à utiliser(PERL) qui détecte les virus, les mets en quarantaine et permet aussi de bloquer certains types de fichiers(mp3)

SPAMS ET VIRUS

Un grand probleme dans notre lutte contre les spams sont les caches des naviguateurs webs qui peuvent être analysés par les moteurs de spam pour y trouver des adresses mails.

Si on recoit un mail provenant d'un relais dont l'adresse est dynamique, il y a de grande chance pour que ca soit un spam. Notons que si on a soi même un serveur SMTP on peut le faire de manière inconsciente (serveur smtp zombie), vu que notre ip change dynamiquement , l'adresse du relais est elle meme dynamique.

Un antivirus sur un serveur mail serait trop souvent réquisitionné, c'est pour ca qu'on préfèe les utiliser sur un pc

TCP-WRAPPERS

Les TCP Wrappers rajoutent une protection supplémentaire aux services lancés par inetd (qui permet de lancer des serveurs démons), c'est en fait un contrôle d'accès, par exemple pour se

connecter avec telnet, il faut d'abord passer le contrôle des TCP Wrappers, une fois passer ce contrôle on peut alors se connecter à telnet, si on ne passe pas le contrôle des TCP Wrappers, la session telnet n'est même pas ouverte. Les TCP Wrappers permettent évidemment de loguer chaque connexion (réussie ou pas). Les fichiers de configuration des TCP Wrappers pour fixer les accès sont /etc/hosts.allow et /etc/hosts.deny. Ces fichiers déterminent qui peut ou ne peut pas accèder aux systèmes (ou du moins aux services lancés par inetd).

La règle est d'interdire tout à tout le monde, puis d'autoriser uniquement certains postes très limités à utiliser vos services.

SÉCURISATION SENDMAIL(PAS ESSENTIEL MAIS POSSIBLE QUESTION)

CRÉATION DU COMPTE D'EXÉCUTION ET MISE EN PLACE DES PERMISSIONS

Il convient de créer en premier lieu le compte de service utilisé par Sendmail pour s'exécuter, et de fixer les droits adéquats sur les fichiers et répertoires utilisés par Sendmail :

- Créer un utilisateur "mail" dans le fichier /etc/passwd et un groupe "mail" dans le fichier /etc/group.

L'utilisateur "mail" ne doit pas disposer de shell (/bin/false comme shell) et son compte doit être désactivé (* dans le champs password), soit une ligne de la forme suivante dans le fichier /etc/passwd :

mail:*:uid:gid:Compte de service Sendmail:/home/mail:/bin/false

- Disposition des droits sur les répertoires

Modifier les permissions du répertoire /var/spool/mail pour que le propriétaire soit root et le groupe mail :

chown root:mail /var/spool/mail

Fixer les droits 1775 sur le répertoire /var/spool/mail (éventuellement 1770 si aucun programme local de messagerie n'accède le répertoire) :

chmod 1775 /var/spool/mail

Tous les répertoires contenus dans /var/spool/mail doivent disposer de droits d'accès 660.

chmod 660 /var/spool/mail/*

Le répertoire /var/spool/mqueue doit appartenir à mail et disposer de droits 700 :

chown mail /var/spool/mqueue ; chmod 600 /var/spool/mqueue.

FORCER SENDMAIL A UTILISER LE SHELL RESTREINT SMRSH

smrsh a pour objectif de remplacer /bin/sh et de contrôler et spécifier la liste explicite des programmes utilisables par Sendmail.

Remarque : ceci nécessite que smrsh soit compilé et installé sur le système (dans /usr/sbin/smrsh).

Il convient donc :

- De déterminer quels programmes sont autorisés pour Sendmail (ex. : /bin/mail, /usr/bin/procmail). Ceci permet aux programmes listés d'être exécutés depuis les fichiers des utilisateurs tels que .forward.
- De préciser à smrsh les programmes autorisés, soit, pour la commande mail :
  cd /etc/smrsh ; ln -s /bin/mail mail
- De configurer Sendmail pour utiliser le shell restreint smrsh :
  Remplacer dans le fichier /etc/mail/sendmail.cf la ligne :
  Mprog, P=/bin/sh, F=lsDFMoqeu9, S=10/30, R=20/40, D=$z:/, T=X-Unix, A=sh -c $u
  Par la ligne :
  Mprog, P=/usr/bin/smrsh, F=lsDFMoqeu9, S=10/30, R=20/40, D=$z:/, T=X-Unix, A=sh -c $u
- Et enfin relancer Sendmail :
  /etc/rc.d/init.d/sendmail restart

Lancer Sendmail depuis Inetd et non comme démon

Pour permettre à Sendmail d'écouter sur le port 25 sans disposer de droits root, il convient de lancer le service non plus comme démon, mais via inetd (le super-démon). Pour celà, ajouter la ligne suivante dans le fichier /etc/inetd.conf :

smtp stream tcp nowait mail /usr/sbin/tcpd /usr/sbin/sendmail -bs

Remarques :

- L'option -bs précise que Sendmail dialogue en SMTP lorsqu'il envoit ou reçoit des messages.

- Lancer Sendmail via Inetd présente également l'avantage de pouvoir utiliser les TCP Wrappers afin de contrôler et filtrer les connexions effectuées avec le serveur de messagerie.

Arrêter le démon Sendmail :

/etc/rc.d/init.d/sendmail stop

Veiller à ne plus permettre à Sendmail de démarrer en tant que démon, par exemple (sous Linux):

chkconfig --del sendmail

Relancer le super-démon inetd :

/etc/rc.d/init.d/inet restart

Scruter la queue de messages régulièrement

Il est maintenant nécessaire de préciser à Sendmail de scruter régulièrement la queue de messages. Il faut donc programmer le démon cron pour lancer périodiquement la vérification de queue par Sendmail.

Par exemple, pour effectuer une scrutation toutes les 5 minutes :

- Éditer la crontab pour le compte de service "mail" :

```
crontab -e -u mail
```

Insérer la ligne suivante (séparateur : tabulation) :

```
*/10	*	*	*	*	/usr/bin/sendmail -q
```

Dernière étape : modifier l'UID et le GID de Sendmail

Sendmail doit être exécuté avec l'UID "mail" et le GID "mail".

Pour cela, ajouter les lignes suivantes dans le fichier /etc/sendmail.mc :

```
define('confTEMP_FILE_MODE','0660')dnl
define('ALIAS_FILE','/etc/mail/aliases')
```

Pour que les changements soient effectués, exécuter les commandes suivantes :

```
m4 /etc/sendmail.mc > /etc/sendmail.cf
/etc/rc.d/init.d/inet restart
```

Changer les permissions sur les répertoires liés à Sendmail de la manière suivante :

```
chown mail:mail /usr/sbin/sendmail
chmod 6555 /usr/sbin/sendmail
chown mail /var/spool/mqueue/*
chgrp mail /etc/aliases.db
chmod 664 /etc/aliases.db
```