http://clx.asso.fr/spip/?HOW-TO-Les-processus-demons-et-les

Comment sécuriser un peu plus sa machine

HOW-TO : Les processus démons et les systÚmes de chrootage

- Documentations - Technique -

Date de mise en ligne : lundi 4 novembre 2002

L'idée de savoir sécuriser un peu plus une machine Unix en permettant le moins d'accÚs possible à travers un service ouvert, et ce grâce à une commande, un peu de méthode et un peu d'imagination.

Apportez toutes vos suggestions d'ajouts et/ou corrections à Jonathan A. Zdziarski

Table des matières

Partie I : Introduction au chrootage

1.1 Qu'est ce le chrootage ?

La commande/fonction chroot est une abréviation de "change root" ("changer la racine"), et sert à changer la racine du système de fichiers pour l'environnement auquel elle est appliquée. Ceci signifie que la racine initiale (/), dans tous les noms de chemin, est définie relativement au chemin chrooté. Par exemple, si un fichier s'appelait : `/home/jonz/hello.txt` sur le système, et si je chroote `/home/jonz` ,
le fichier se nommerait alors, dans l'environnement chrooté, quelque chose comme : `/hello.txt`

Le but du chrootage consiste à créer (théoriquement)
une "prison" impénétrable (ou cellule) protégeant de la lecture ou de la modification tous fichiers en dehors de l'environnement chrooté. Dans l'exemple ci-dessus, je ne pourrais accéder à aucun fichier en dehors de `/home/jonz` , puisque `/` correspond maintenant à `/home/jonz` . Le chrootage est généralement employé pour emprisonner des utilisateurs dans des environnements à
utilisateurs multiples afin de protéger les fichiers propres au système.

Le chrootage peut également être utilisé pour emprisonner
les démons du système afin que ces derniers ne deviennent pas des cibles faciles pour d'éventuelles intrusions. Si une intrusion exploite
une vulnérabilité dans un démon d'un système chrooté, sa capacité à affecter des fichiers en dehors de la prison,
ou à obtenir un shell sera sensiblement diminuée. Une des principales raisons à cela est que le shell ne donne pas d'accès à l'environnement complet. Ainsi même si l'intrus injecte du code dans la mémoire protégée ("Démonte" la pile d'appel), il n'y a aucun shell à exécuter. Beaucoup de personnes ont prétendu pouvoir s'introduire par effraction dans une prison chrootée, mais dans beaucoup de cas il ne s'agissait que d'un simple shell (ce n'est pas le cas que nous allons étudier). S'introduire par effraction dans une prison abritant un démon d'un environnement chrooté est pour le moins, extrêmement difficile.

1.2 Quand est-il approprié de chrooter un processus système ou un démon ?

Chrooter un démon est une méthode pratique pour ajouter une couche supplémentaire de sécurité à votre système.

Beaucoup de processus du système et applications tierces ont déjà quelques garde-fous contre une exploitation de leurs vulnérabilités potentielles.

Beaucoup d'outils ont maintenant la capacité de fonctionner sans les droits administrateurs, rendant ainsi l'attaque en tant que super-utilisateur (root) plus délicate pour les intrus.

Des couches de sécurité pour les réseaux comme des pare-feux, des encapsulages TCP, des filtres, at autres s'ajoutent également à la sécurité globale d'un système. Comme tous ces derniers, le chrootage est approprié pour la plupart des implémentations, encore faut-il que cela ne bride pas les fonctionnalités du système.

1.3 Tous les démons peuvent-ils être chrootés ?

Techniquement vous pouvez chrooter tout ce que vous voulez même la bicyclette de votre oncle Jules, mais dans certains cas le chrootage n'est pas toujours possible sans "casser" quelque chose, ou dans d'autres circonstances,

sans une configuration très complexe, qui n'en vaut finalement pas la peine.

Quelques démons ne peuvent pas fonctionner correctement dans un environnement chrooté du fait de la grande complexité de leurs fonctions.

Par exemple, sendmail doit pouvoir accéder aux répertoires locaux des utilisateurs à la recherche de leurs fichiers .forward. Il n'y a aucun moyen pratique de chrooter sendmail sans créer un "miroir" complexe et consommateur de temps. C'est pourquoi sendmail possède une solution de rechange, smrsh (shell restreint de sendmail â€" *sendmail restricted shell*).

Une majorité de démons système, peut cependant être chrooté sans risque ni trop d'effort.

1.4 Le chrootage affectera-t-il mes utilisateurs ?

Si le chrootage est fait correctement, vos utilisateurs ne devraient noter aucune différence dans le comportement du système. Le chrootage en soi n'affectera pas directement vos utilisateurs ou ne changera pas votre système d'exploitation. Le système existant est généralement laissé intact, alors que de petites "prisons" sont créées afin de fournir de nombreux services. Gardez à l'esprit également, nous ne parlons pas de tirer profit des versions chrootées existantes pour les démons ftp ou ssh. C'est un concept semblable, mais pourtant différent de celui que nous allons aborder ici. Nous parlerons du chrootage des démons système qui continueront à fonctionner de façon toute aussi transparente mais avec une couche de sécurité en plus.

1.5 Qu'impose le chrootage ?

Le chrootage de n'importe quel démon est un processus. Il implique généralement les étapes suivantes pour produire un démon chrooté viable :

– Créer un répertoire "prison",
– Copier le démon et tous les fichiers requis,
– Copier les bibliothèques système requises,
– Changer tous scripts de démarrage pour lancer le démon dans son nouvel environnement.

Les étapes 2 et 3 sont celles qui prennent généralement le plus de temps. L'étape 2, en particulier, requiert parfois une certaine imagination. Par exemple, un logiciel de courrier doit accéder à certains dossiers sur le système. Copier les dossiers de messagerie dans la prison ne produira pas le résultat désiré, nous devons alors créer une prison "autour" de ces dossiers.

Nous entrerons dans le vif du sujet et les manipulations complexes plus tard.

Partie II : Prérequis

2.1 Pouvons-nous chrooter ce démon ?

Avant même de créer un répertoire ou de copier un simple
fichier, nous devons nous poser quelques questions importantes. La première est de savoir si le démon qui nous intéresse peut être chrooté.

Nous allons travailler avec trois exemples de démons systèmes ci-dessous, et poserons les mêmes questions principales :
– Le démon accède-t-il à des fichiers présents dans un endroit
unique ?
– Est-il viable de copier les fichiers une fois pour toute, ou périodiquement en utilisant cron ?
– Est-il prudent de créer une prison "autour" d'un ensemble spécifique de fichiers ?

Si vous répondez 'NON' à chacune de ces trois questions,
il se pourrait que ce ne soit pas une bonne idée de chrooter ce démon.

Mais jetons un oeil aux exemples suivants :

Qpopper de Qualcomm

Le démon accède-t-il à des fichiers présents dans un endroit unique ?

Principalement, un logiciel de courrier doit accéder aux dossiers des utilisateurs. Il y a quelques autres dossiers tels que ceux de /etc qui sont exigés pour effectuer l'authentification. La réponse à cette question est donc 'OUI' car une majorité des dossiers requis sont présents dans un seul endroit (/var/mail).

Est-il viable de copier les fichiers une bonne fois pour toute, ou périodiquement en utilisant cron ?

Copier les répertoires de messagerie n'est jamais une bonne idée, et les copier une seule fois finira par corrompre les boîtes aux lettres des utilisateurs. Ainsi, ce n'est pas du tout une bonne idée de copier ces fichiers. Il est cependant prudent de recopier
les fichiers et répertoires de /etc ainsi que tout autre utilisé dans cet exemple, une fois par jour, ou plus ou moins fréquemment selon les besoins. Puisque nous essayons de protéger ces fichiers, nous ne voulons pas donner d'accès à l'intrus potentiel pour les originaux.

Est-il prudent de créer une prison "autour" d'un ensemble spécifique de fichiers ?

Il y a deux solutions de rechange pour traiter les dossiers d'une prison. Soit on déplace ou copie les fichiers dans la prison, ou on construit la prison autour des fichiers. Dans le cas du courrier électronique, copier les fichiers dans les deux sens toutes les x minutes est extrêmement consommateur
de ressources, et il est tout simplement inefficace d'essayer de maintenir un répertoire plein de liens symboliques. Il est bien plus efficace de forcer les logiciel à utiliser les dossiers réels dans leur endroit original. Notre réponse est ici encore 'OUI '.

RPCBIND de Sun MicroSystem.

Le démon accède-t-il à des dossiers présents dans un endroit unique ?

Le démon accède principalement à plusieurs petits fichiers
dans /etc.

Est-il viable de copier les fichiers une bonne fois pour toute, ou périodiquement en utilisant cron ?

La plupart des dossiers consultés sont petits et rarement
mis à jour. Il serait tout à fait viable de copier ces fichiers dans un répertoire différent une fois par jour.

Est-il prudent de créer une prison "autour" d'un ensemble spécifique de dossiers ?

A la différence de notre logiciel de courrier, qui accède
principalement aux mails , /etc contient plusieurs des fichiers système critiques que nous essayons de nous protéger. Ce serait une très mauvaise idée de créer une prison autour de ces derniers et ferait perdre le bénéfice premier du chrootage. De plus, cela risque donner la possibilité à un intrus d'accéder au moindre fichier du répertoire /etc. Ainsi, la bonne idée consiste à chrooter CE démon dans une prison totalement séparée, et en y copiant simplement les fichiers utilisés par RPCBind.

Sendmail

Le démon accède-t-il à des dossiers ou fichiers présents dans un endroit unique ?

Non. Sendmail doit avoir accès à plusieurs fichiers situés dans des répertoires différents. Il s'agit aussi bien de répertoires
locaux que de fichiers utilisateurs. Simultanément, sendmail doit également accéder à la queue d'envoi de courrier (/var/spool/mqueue) et à des répertoires de courrier spécifiques aux utilisateur (/var/mail).

Est-il viable de copier les fichiers une bonne fois pour toute, ou périodiquement en utilisant cron ?

Afin de créer un environnement chrooté contenant tous les
fichiers nécessaires au fonctionnement de sendmail, des répertoires utilisateurs devraient être dupliqués dans un endroit sécurisé. Ce dernier sert à contenir les fichiers nécessaires à la distribution et à la manipulation de courrier. Ceci exigerait également la création de nouveaux répertoires et de la suppression des plus anciens après qu'ils aient été modifiés par l'utilisateur. Simultanément, sendmail doit également accéder aux répertoires de la queue d'envoi et aux répertoires de courrier des utilisateurs.

Il n'est viable de copier aucun de ces derniers que ce soit dans un sens ou dans l'autre.

Est-il prudent de créer une prison "autour" d'un ensemble spécifique de fichiers ?

Attardons-nous sur cette question. Nous pourrions créer une
prison autour des dossiers, mais alors nous perdrions la possibilité d'écrire dans la queue d'envoi de courrier. Nous pourrions emprisonner /var, permettant ainsi l'accès à tous les deux (/var/mail et /var/spool/mqueue) mais nous laisserions alors l'environnement chrooté accéder aux dossiers potentiellement critiques. Ni l'une ni l'autre des solutions envisagées n'aborde l'autre question, à savoir la duplication des répertoires utilisateurs. Il ne semble y avoir aucun ensemble de fichiers pour lequel nous pourrions créer une prison.

Sendmail, comme nos réponses le suggèrent, n'est pas un bon candidat pour le chrootage . Bien que cela puisse être possible, l'installation serait longue et fastidieuse. Répondre exactement aux contraintes de sécurité accompagnant la documentation de sendmail est déjà une bonne idée.

2.2 Introduction aux commandes truss, lsof et ldd

truss (ou strace)

Si vous tapez `man truss` sous Solaris, vous verrez que truss est un outil conçu pour tracer des appels et signaux système. Strace est un outil semblable, sous Linux. Les appels système incluent des appels aux bibliothèques et aux fichiers en cours d'accès. Truss est une superbe moyen de découvrir les ressources utilisées par un démon particulier. Pour lancer truss, tapez simplement 'truss' suivi de la commande servant habituellement à démarrer le démon. La sortie ressemblera à ceci :

```
open("/usr/lib/libc.so.1 ", O_rdonly)= 3
```

Le programme a ouvert la bibliothèque libc.

```
open64("/usr/local/etc/my.config ", O_wronly|O_creat|O_trunc, 0666) = 3
```

Le programme a ouvert /usr/local/etc/my.config adin d'y écrire des informations.

Exécuter truss sur un démon devrait donner immédiatement
une longue liste de lignes. Il est intéressant d'essayer les différentes options et commandes du démon pour voir les nouveaux fichiers ouverts avec truss. Sans truss, vous seriez obligé de casser la prison formée par l'environnement chrooté à chaque fois que manquerait un fichier.

lsof

lsof est un autre outil très utile pour trouver les ressources utilisées par un démon. Il affiche les fichiers ouverts, les connexions réseaux établies, et plus encore. Le programme lsof n'est pas installé systématiquement avec le système, c'est un outil externe. Il peut être téléchargé depuis plusieurs sites comme www.sunfreeware.com. Les sources sont disponibles sur les ftp anonymes comme à l'adresse
ftp://vic.cc.purdue.edu/pub/tools/unix/lsof.

ldd

ldd est outil très pratique pour déterminer le nom des bibliothèques nécéssaires à l'exécution de votre programme. L'outil ldd se limite à lister les "bibliothèques dynamiques" [1]. Utilisez ldd suivi du chemin et et du nom de fichier du démon que vous voulez chrooter. Cela vous donnera la liste des bibliothèques à copier dans la prison.

2.3 Déterminer les dépendances des différents fichiers

Le premier exercice que nous avons abordé à la section 2.1 nous a fait réfléchir sur les dépendances (fichiers, dossiers, librairies, etc.) du démon. Certaines sont installables dans d'autres dossiers alors que pour d'autres nous

devrons établir une prison tout autour. Seul deux des trois exemples de la section étaient viables (Qpopper de Qualcomm et RPCBind de Sun). Ils sont de très bon candidats pour le chrootage et tous deux ont connu des failles de sécurité non négligeables par le passé. Maintenant établissons la liste des fichiers que nous aurons besoin de déplacer ou de recompiler pour ces deux exemples. Il est tout à fait possible de rater quelques fichiers puisque nous trouverons toujours une solution pour les inclure dans la prison avant la mise en route de ce nouveau service. Il y a plusieurs façons de déterminer les fichiers utiles. Ceci inclus, mais n'est pas exhaustif :
– Penser aux évidences (et oui, un logiciel de mail accède
aux répertoires de mails),
– Examiner les fichiers de configuration,
– Utiliser des outils comme 'lsof' ou 'truss',
– Arrêter le démon pour le placer dans sa prison.

Il est important de tenir compte du maximum de fichiers possibles. D'après ces deux exemples, nous pouvons établir une liste exhaustive de ce dont nous allons avoir besoin :

Qpopper de Qualcomm :

/var/mail/* (dans la prison)

Les fichiers suivants, provenant de /etc, seront recopiés périodiquement

```
hosts.allow hosts.deny netconfig nsswitch.conf passwd resolv.conf services shadow
```

Sun RPCBIND

Les fichiers suivants, provenant de /etc, seront recopiés périodiquement

```
hosts netconfig nsswitch.conf resolv.conf services
```

La zone horaire, de `/usr/local/zoneinfo`.

Un répertoire temporaire, /tmp.

2.4 Créer une stratégie

Une fois que que nous avons une idée précise de la liste de fichiers qui seront utilisés, voyons voir comment nous pouvons les placer dans notre prison. Les copierons-nous ? Combien de fois ? Une seule fois ou quotidiennement ? La prison peut elle être construite autour ? Établir une stratégie vous montrera les fichiers devant être copiés, ceux pour lesquels il faudra construire une prison autour, et vous aurez alors une idée de l'emplacement futur de la prison. **Il est impératif** de ne pas construire la prison autour de fichiers critiques , comme /etc, ce qui élimine l'intérêt de notre prison. Si vous devez absolument construire une prison autour de quoique ce soit, il faut le faire autour d'un répertoire non critique pour le système, afin d'éviter qu'un intrus éventuel exploite une faille de sécurité.

Si vous tracez un cercle sur le papier, vous pouvez positionner les fichiers en dehors de la prison à l'extérieur du cercle. Tracez des lignes dans le cercle avec des annotations pour montrer ceux qui seront déplacés ou copiés, en précisant la fréquence des opérations.
Ceci pour montrer la taille de la future prison (la racine du système de fichiers virtuel) et vous donnera de bonnes bases de travail quand il sera temps de construire la prison.

2.5 Déterminer les dépendances des bibliothèques

Tout le code du programme que le démon exécutera n'est pas nécessairement dans un seul programme. Beaucoup de bouts de code partagés sont réutilisés sous la forme de bibliothèques. Celles-ci sont requises par le programme pour s'exécuter correctement. Dans un environnement chrooté, le programme n'aura pas accès au chemin original de ces bibliothèques. Il faudra d'abord les copier dans la prison pour le faire fonctionner. Le meilleur moyen de déterminer quelles seront les bibliothèques utilisées par votre démon est d'utiliser la commande 'ldd'. le programme ldd listera les bibliothèques dynamiques (ou bibliothèques partagées) requises pour l'exécuter. Il faut faire très attention à cette partie, il faudra les copier dans le prison.

Documentez les bibliothèques utilisées comme vous l'avez fait pour les fichiers. Vous savez déjà comment il faudra les copier dans la prison (avec cp, et une seule fois). C'est peut être une bonne idée de garder cette information pratique sous le coude dans le cas d'une mise à jour ou afin d'appliquer un patch du système d'exploitation. Cela pourra vous être utile pour copier les bibliothèques mises à jour dans la prison également. Une solution pourrait être de les recopier via cron une fois par mois.

Dans nos exemples, nous avons extrait les bibliothèques suivantes en utilisant truss :

Qpopper de Qualcomm

```
/usr/lib/libnsl.so.1 /usr/lib/libsocket.so.1 /usr/lib/libresolv.so.2 /usr/lib/libmail.so.1
/usr/lib/librt.so.1 /usr/lib/libcrypt_i.so.1 /usr/lib/libc.so.1 /usr/lib/libdl.so.1 /usr/lib/libmp.so.2
/usr/lib/libaio.so.1 /usr/lib/libgen.so.1 /usr/lib/libpthread.so.1 /usr/lib/libthread.so.1
```

Sun RPCBIND

```
/usr/lib/libsocket.so.1 /usr/lib/libnsl.so.1 /usr/lib/libdl.so.1 /usr/lib/libc.so.1 /usr/lib/libmp.so.2
```

2.6 Trouver le meilleur emplacement pour la prison

Vous avez désormais toutes les informations nécessaires pour savoir où installer votre prison. Si votre prison est 'construite autour' de fichiers existants (comme pour l'exemple de Qpopper), vous devrez être le plus proche possible de ce répertoire. Si ce n'est pas le cas, choisissez un emplacement arbitraire pourvu qu'il y ait suffisamment d'espace disque. Nous allons utiliser `/var/mail` pour popper et `/var/rpcbind` pour Sun RPCBIND. Souvenez-vous que vos programmes ne peuvent savoir qu'ils sont emprisonnés et vont donc essayer d'accéder à des fichiers où le démon s'attend à les trouver, que leur chemin soit relatif ou liés symboliquement. Vous êtes prêt pour passer à la troisième partie !

Partie III : L'installation de l'environnement chrooté

3.1 Créer une prison vide

La première étape pour créer une prison vide est de créer un ensemble de répertoires pour y mettre votre démon, ses fichiers de configuration et ses bibliothèques. Tout d'abord, voyons le cas de Sun RPCBIND. Nous avons choisi `/var/rpcbind` pour être la racine de notre prison. Voyons comment préparer notre arborescence :

```
mkdir /var/rpcbind mkdir /var/rpcbind/dev mkdir /var/rpcbind/etc mkdir /var/rpcbind/tmp mkdir -p
/var/rpcbind/usr/lib
```

Dès que le démon sera chrooté, il devra accéder à certains devices dans /dev, certains fichiers de configuration de /etc, son propre répertoire temporaire, et son répertoire
de bibliothèques. Tout devra y être quand vous aurez fini.

En ce qui concerne notre exemple avec Qpopper. Comme la prison de Qpopper doit être construite autour du répertoire des boîtes aux lettres, la prison doit démarrer dans /var/mail. Nous devons continuer et créer les répertoires suivant :

```
mkdir /var/mail/dev mkdir /var/mail/etc mkdir -p /var/mail/usr/lib  mkdir -p /var/mail/usr/local/lib mkdir
-p /var/mail/usr/sbin mkdir -p /var/mail/usr/share/lib/zoneinfo/US
```

Mais ce n'est pas encore terminé. Notre logiciel
s'attend à ce que les mails soient dans /var/mail. Comme il est chrooté, / est en fait /var/mail ... et lorsqu'il accède à /var/mail, il ne trouve pas son fichier ! Comment résoudre se problème ? Nous devons créer un lien symbolique de /var/mail/var/mail vers /var/mail qui est alors /...

```
mkdir /var/mail/var ln -s / /var/mail/var/mail
```

Maintenant, lorsque popper chrooté accèdera à /var/mail, il sera redirigé vers /, qui est bien le vrai
/var/mail !

Le résultat final ressemblera alors à ceci :

Sun RPCBIND

```
drwxr-xr-x   7 root     other        512 Aug  1 18:31 ./ drwxr-xr-x  34 root     sys          512 Aug  1
18:07 ../ drwxr-xr-x   2 root     other        512 Aug  1 18:16 dev/ drwxr-xr-x   2 root     other
512 Aug  1 18:10 etc/ drwxr-xr-x   2 root     other        512 Aug  1 18:31 tmp/ drwxr-xr-x   6 root
other        512 Jul 28 15:55 usr/ drwxr-xr-x   3 root     other        512 Aug  1 18:19 var/
```

Qpopper

```
drwxrwxrwt   7 root     mail         512 Oct 18 19:36 ./ drwxr-xr-x  34 root     sys          512 Aug  1
18:07 ../ drwxr-xr-x   2 root     other        512 Jul 29 13:33 dev/ drwxr-xr-x   2 root     other
512 Jul 28 16:18 etc/              - Quelques boîtes aux lettres mélangées par ici -  drwxr-xr-x   6 root
 other        512 Jul 28 15:55 usr/ drwxr-xr-x   2 root     other        512 Aug  1 18:06 var/
```

3.2 Copier les programmes et fichiers de configuration, configurer les tâches automatiques

Maintenant que notre arborescence est établie, copions-y nos programmes et fichiers de configuration. Comme nous démarrerons RPCBind via les scripts d'initialisation, il n'est pas nécessaire de copier le binaire actuel. Cependant, nous devrons y copier Qpopper pour qu'il puisse être démarré via inetd. Si inetd cherche le fichier dans /usr/bin, vous devrez créer /usr/bin dans la prison (/var/mail/usr/bin) et y copier l'exécutable du démon. Copiez-y tous les fichiers de /etc identifiés tout à l'heure en préservant les permissions. Pour les fichiers devant être mis à jour périodiquement, créez les tâches 'cron' adéquates. Une fois terminé, votre structure ressemblera à ceci :

Sun RPCBIND

```
% ls -l /var/rpcbind/etc  -r--r--r--   1 root     other         90 Jul 28 16:18 hosts -rw-r--r--   1 root
```

```
 other       1239 Jul 28 16:09 netconfig -rw-r--r--   1 root     other        835 Jul 28 15:32 nsswitch.conf
-rw-r--r--   1 root     other        140 Jul 28 16:14 resolv.conf -r--r--r--   1 root     other       3649
Jul 28 16:14 services
```

Qpopper

```
% ls -l /var/mail/etc  -r--r--r--   1 root     other         90 Jul 28 16:18 hosts -rw-------   1 root
other         73 Oct 18 19:15 hosts.allow -rw-------   1 root     other          9 Jul 28 15:58 hosts.deny
-rw-r--r--   1 root     other       1239 Jul 28 16:09 netconfig -rw-r--r--   1 root     other        835 Jul
28 15:32 nsswitch.conf -r--r--r--   1 root     other        815 Oct 18 19:15 passwd -rw-r--r--   1 root
other        140 Jul 28 16:14 resolv.conf -r--r--r--   1 root     other       3649 Jul 28 16:14 services
-r--------   1 root     other        502 Oct 18 19:15 shadow
```

Les lignes cron utiles pourraient ressembler à ceci. Adaptez-les en fonction de votre cas particulier...

```
0,30 * * * * cp -p /etc/passwd /var/mail/etc/passwd 0,30 * * * * cp -p /etc/shadow /var/mail/etc/shadow 0 0
* * * cp -p /etc/hosts.* /var/mail/etc/ 0 0 0 * * cp -p /etc/services /var/rpcbind/etc 0 0 0 * * cp -p
/etc/resolv.conf /var/mail/etc
```

3.3 Copier les bibliothèques

Vous vous souvenez de notre topo au sujet des bibliothèques ? Nous devons les copier dans le chemin relatif dans la prison. La plupart des bibliothèques se trouvent dans /usr/lib, et parfois dans /usr/local/lib. De même, pour les fichiers de configuration, copiez les librairies en conservant leurs permissions. Vous devriez aboutir à quelque chose comme ceci selon votre système d'exploitation :

```
% ls -l /var/rpcbind/usr/lib  -rwxr-xr-x   1 root     other     200292 Jul 28 15:27 ld.so.1* -rwxr-xr-x   1
root     other      41628 Jul 28 15:28 libaio.so.1* -rwxr-xr-x   1 root     other     938940 Jul 28 15:28
libc.so.1* -rwxr-xr-x   1 root     other      15616 Jul 28 15:27 libcrypt_i.so.1* -rwxr-xr-x   1 root
other       4448 Jul 28 15:28 libdl.so.1* -rwxr-xr-x   1 root     other      40540 Jul 28 15:28 libgen.so.1*
-rwxr-xr-x   1 root     other      29548 Jul 28 15:27 libmail.so.1* -rwxr-xr-x   1 root     other      19584
Jul 28 15:28 libmp.so.2* -rwxr-xr-x   1 root     other     730672 Jul 28 15:27 libnsl.so.1* -rwxr-xr-x   1
root     other      35308 Jul 28 15:57 libpthread.so.1* -rwxr-xr-x   1 root     other     326336 Jul 28
15:27 libresolv.so.2* -rwxr-xr-x   1 root     other      39048 Jul 28 15:27 librt.so.1* -rwxr-xr-x   1 root
   other      65876 Jul 28 15:27 libsocket.so.1* -rwxr-xr-x   1 root     other     166624 Jul 28 15:58
libthread.so.1* -rwxr-xr-x   1 root     other      19648 Jul 28 16:17 nss_dns.so.1* -rwxr-xr-x   1 root
other      38832 Jul 28 15:33 nss_files.so.1* -rwxr-xr-x   1 root     other      38292 Jul 28 15:33
nss_nis.so.1* -rwxr-xr-x   1 root     other      12284 Aug  1 18:15 straddr.so*
% ls -l /var/mail/usr/lib /var/mail/usr/local/lib  -rwxr-xr-x   1 root     other     200292 Jul 28 15:27
ld.so.1* -rwxr-xr-x   1 root     other      41628 Jul 28 15:28 libaio.so.1* -rwxr-xr-x   1 root     other
 938940 Jul 28 15:28 libc.so.1* -rwxr-xr-x   1 root     other      15616 Jul 28 15:27 libcrypt_i.so.1*
-rwxr-xr-x   1 root     other       4448 Jul 28 15:28 libdl.so.1* -rwxr-xr-x   1 root     other      40540
Jul 28 15:28 libgen.so.1* -rwxr-xr-x   1 root     other      29548 Jul 28 15:27 libmail.so.1* -rwxr-xr-x   1
root     other      19584 Jul 28 15:28 libmp.so.2* -rwxr-xr-x   1 root     other     730672 Jul 28 15:27
libnsl.so.1* -rwxr-xr-x   1 root     other      35308 Jul 28 15:57 libpthread.so.1* -rwxr-xr-x   1 root
other     326336 Jul 28 15:27 libresolv.so.2* -rwxr-xr-x   1 root     other      39048 Jul 28 15:27
librt.so.1* -rwxr-xr-x   1 root     other      65876 Jul 28 15:27 libsocket.so.1* -rwxr-xr-x   1 root
other     166624 Jul 28 15:58 libthread.so.1* -rwxr-xr-x   1 root     other      19648 Jul 28 16:17
nss_dns.so.1* -rwxr-xr-x   1 root     other      38832 Jul 28 15:33 nss_files.so.1* -rwxr-xr-x   1 root
other      38292 Jul 28 15:33 nss_nis.so.1*
```

3.4 Créer les devices

Voici la partie la plus délicate. Une fois le démon chrooté, il ne pourra plus accéder au répertoire habituel /dev, qui inclus tous les *devices* (périphériques indispensables comme peuvent l'être /dev/null, /dev/log, etc. Isoler les *devices* dont votre démon aura besoin n'est pas évident. Vous pouvez en trouver la plupart en utilisant truss ou parfois avec un `strings | grep dev`, mais généralement, vous devrez observer les messages d'erreurs qui indiqueront le *device* manquant.

Voici les *devices* qui devraient être ajoutés dans la prison, pour n'importe quel démon :

```
/dev/conslog /dev/log /dev/msglog /dev/null /dev/tcp /dev/ticlts /dev/ticots /dev/ticotsord /dev/udp /dev/zero
```

Voici comment créer vos *devices* (il faut le faire pour chaque *device*) :

1. saisir `'ls -lL /dev/{{devicename}}'`. La sortie ressemblera alors à ceci :

```
crw-rw-rw-  1 root     sys 13, 2 Oct 18 19:56 /dev/null
```

Le chiffre en rouge désigne le nombre 'majeur'. Le nombre en bleu est le nombre 'mineur'. La première lettre, en vert, pourra être un 'c' ou un 'b', pour respectivement un *device* de type caractère ou de type bloc. Vous pouvez créer votre noeud à l'aide de ces informations.

2. Déplacez-vous dans le répertoire dev de la prison (c.à.d. /var/mail/dev ou /var/rpcbind/dev). Utilisez la commande 'mknod' pour créer le *device*. Par exemple : `mknod null c 13 2` pour créer un des *devices* listés dans notre exemple. Les machines ont souvent des nombres mineurs et majeurs différents, il convient donc d'y faire très attention.

3. Au besoin, utilisez chown et chmod, prenez garde à ce que les *devices* aient exactement les mêmes permissions que les originaux.

Pour notre exemple de Sun RPCBIND, nous avons fini par découvrir une fois arrêté qu'il fallait aussi créer un répertoire nommé rpc_door avec le sticky bit. Il se peut que ce ne soit pas le cas sur votre système, mais devrez probablement faire quelque chose ressemblant à ceci :

```
mkdir -p /var/rpcbind/var/run/rpc_door chmod +t /var/rpcbind/var/run/rpc_door
```

C'en est fini du baratin.

3.5 Modifier les scripts de démarrage

On y est presque ! Si vous avez oublié quelque chose, vous le verrez probablement en démarrant le démon, la première fois. Vous l'avez testé n'est ce pas ? Ah oui c'est vrai, on ne vous a pas dit comment ! Bon, avant de pouvoir tester correctement la plupart des démons, il faudra arrêter la version non chrootée. Une fois le démon en production tué, exécutez cette commande :

```
/usr/sbin/chroot /var/rpcbind /usr/sbin/rpcbind
```

Si tout va bien, ça devrait fonctionner. Si vous avez des erreurs du style fichiers ou bibliothèques manquantes, lancez truss pour déterminer lesquels et les copier dans le nouveau répertoire. Une fois que tout est bon, commentez simplement l'ancien script de démarrage et remplacez-le par la version chrootée.

Comment chrooter depuis inetd

Qpopper de Qualcomm démarre via inetd, et parfois nécessite une ligne de commande spécigique dans /etc/inetd.conf pour s'exécuter dans l'environnement chrooté. Pour cela, utilisez une ligne semblable à celle-ci :

```
pop3    stream  tcp     nowait  root    /usr/sbin/chroot          chroot /var/mail /usr/sbin/in.pop3
```

Ou si vous utilisez les TCP wrappers, vous devrez leurrer inetd en lui faisant toujours croire que c'est in.pop3 qui démarre (ou ce que vous êtes en train d'étudier). Pour cela, créez un lien symbolique entre le service à chrooter et insérez-le dans inetd.conf. Par exemple :

```
lrwxrwxrwx   1 root     other         6 Jul 28 15:42 /usr/sbin/in.pop3 -> chroot*     pop3    stream  tcp
 nowait  root    /usr/sbin/tcpd          in.pop3  /var/mail /usr/sbin/in.pop3
```

/usr/sbin/in.pop3 est *maintenant* /usr/sbin/chroot (lié symboliquement), et de fait quand inetd démarre le service in.pop3, chroot démarre alors avec les paramètres '/var/mail /usr/sbin/in.pop3' ce qui démarrera la version chrootée de in.pop3 située dans /var/mail. Quand la copie chrootée est démarrée par chroot, le service est correctement configuré.

3.6 Le produit final

Pour finir, notre projet ressemblera à ceci :

Sun RPCBIND

```
d none /var/rpcbind 0755 root other d none /var/rpcbind/dev 0755 root other c none /var/rpcbind/dev/conslog
21 0 0666 root other c none /var/rpcbind/dev/log 21 5 0640 root other c none /var/rpcbind/dev/msglog 97 1
0600 root other c none /var/rpcbind/dev/null 13 2 0666 root other c none /var/rpcbind/dev/udp 41 0 0666 root
other c none /var/rpcbind/dev/tcp 42 0 0666 root other c none /var/rpcbind/dev/ticlts 105 2 0666 root other
c none /var/rpcbind/dev/ticotsord 105 1 0666 root other c none /var/rpcbind/dev/ticots 105 0 0666 root other
d none /var/rpcbind/var 0755 root other d none /var/rpcbind/var/run 0755 root other d none
/var/rpcbind/var/run/rpc_door 1777 root root d none /var/rpcbind/tmp 0755 root other d none /var/rpcbind/usr
0755 root other d none /var/rpcbind/usr/share 0755 root other d none /var/rpcbind/usr/share/lib 0755 root
other d none /var/rpcbind/usr/share/lib/zoneinfo 0755 root other d none
/var/rpcbind/usr/share/lib/zoneinfo/US 0755 root other f none /var/rpcbind/usr/share/lib/zoneinfo/US/Eastern
0644 root bin d none /var/rpcbind/usr/lib 0755 root other f none /var/rpcbind/usr/lib/ld.so.1 0755 root
other f none /var/rpcbind/usr/lib/libnsl.so.1 0755 root other f none /var/rpcbind/usr/lib/libsocket.so.1
0755 root other f none /var/rpcbind/usr/lib/libresolv.so.2 0755 root other f none
/var/rpcbind/usr/lib/libmail.so.1 0755 root other f none /var/rpcbind/usr/lib/librt.so.1 0755 root other f
none /var/rpcbind/usr/lib/libcrypt_i.so.1 0755 root other f none /var/rpcbind/usr/lib/libc.so.1 0755 root
other f none /var/rpcbind/usr/lib/libdl.so.1 0755 root other f none /var/rpcbind/usr/lib/libmp.so.2 0755
root other f none /var/rpcbind/usr/lib/libaio.so.1 0755 root other f none /var/rpcbind/usr/lib/libgen.so.1
0755 root other f none /var/rpcbind/usr/lib/nss_files.so.1 0755 root other f none
/var/rpcbind/usr/lib/nss_nis.so.1 0755 root other f none /var/rpcbind/usr/lib/libpthread.so.1 0755 root
other f none /var/rpcbind/usr/lib/libthread.so.1 0755 root other f none /var/rpcbind/usr/lib/nss_dns.so.1
```

```
0755 root other f none /var/rpcbind/usr/lib/straddr.so 0755 root other d none /var/rpcbind/usr/sbin 0755
root other f none /var/rpcbind/usr/sbin/rpcbind 0555 root other d none /var/rpcbind/usr/local 0755 root
other d none /var/rpcbind/usr/local/sbin 0755 root other d none /var/rpcbind/usr/local/lib 0755 root other d
none /var/rpcbind/etc 0755 root other f none /var/rpcbind/etc/nsswitch.conf 0644 root other f none
/var/rpcbind/etc/netconfig 0644 root other f none /var/rpcbind/etc/services 0444 root other f none
/var/rpcbind/etc/resolv.conf 0644 root other f none /var/rpcbind/etc/hosts 0444 root other
```

Qpopper de Qualcomm

```
d none /var/mail 1777 root mail d none /var/mail/usr 0755 root other d none /var/mail/usr/share 0755 root
other d none /var/mail/usr/share/lib 0755 root other d none /var/mail/usr/share/lib/zoneinfo 0755 root other
d none /var/mail/usr/share/lib/zoneinfo/US 0755 root other f none
/var/mail/usr/share/lib/zoneinfo/US/Eastern 0644 root bin d none /var/mail/usr/lib 0755 root other f none
/var/mail/usr/lib/ld.so.1 0755 root other f none /var/mail/usr/lib/libnsl.so.1 0755 root other f none
/var/mail/usr/lib/libsocket.so.1 0755 root other f none /var/mail/usr/lib/libresolv.so.2 0755 root other f
none /var/mail/usr/lib/libmail.so.1 0755 root other f none /var/mail/usr/lib/librt.so.1 0755 root other f
none /var/mail/usr/lib/libcrypt_i.so.1 0755 root other f none /var/mail/usr/lib/libc.so.1 0755 root other f
none /var/mail/usr/lib/libdl.so.1 0755 root other f none /var/mail/usr/lib/libmp.so.2 0755 root other f none
/var/mail/usr/lib/libaio.so.1 0755 root other f none /var/mail/usr/lib/libgen.so.1 0755 root other f none
/var/mail/usr/lib/nss_files.so.1 0755 root other f none /var/mail/usr/lib/nss_nis.so.1 0755 root other f
none /var/mail/usr/lib/libpthread.so.1 0755 root other f none /var/mail/usr/lib/libthread.so.1 0755 root
other f none /var/mail/usr/lib/nss_dns.so.1 0755 root other d none /var/mail/usr/sbin 0755 root other f none
/var/mail/usr/sbin/in.pop3 0755 root other d none /var/mail/usr/local/lib 0755 root other d none
/var/mail/etc 0755 root other f none /var/mail/etc/passwd 0444 root other f none /var/mail/etc/shadow 0400
root other f none /var/mail/etc/hosts.allow 0600 root other f none /var/mail/etc/nsswitch.conf 0644 root
other f none /var/mail/etc/hosts.deny 0600 root other f none /var/mail/etc/netconfig 0644 root other f none
/var/mail/etc/services 0444 root other f none /var/mail/etc/resolv.conf 0644 root other f none
/var/mail/etc/hosts 0444 root other d none /var/mail/var 0755 root other s none /var/mail/var/mail=/ d none
/var/mail/dev 0755 root other c none /var/mail/dev/udp 41 0 0666 root other c none /var/mail/dev/null 13 2
0666 root other c none /var/mail/dev/conslog 21 0 0666 root other c none /var/mail/dev/log 21 5 0640 root
other c none /var/mail/dev/msglog 97 1 0600 root other
```

Les démons que j'ai personnellement [2] chrooté
sans problème :
– Qpopper
– RPCBind
– Named (c'est beaucoup mieux que la version de base)
– Stunnel
– Apache
– Nscd
– SecurID Server
– RADIUS
– Squid
– syslogd

Les démons théoriquement chrootables, mais que je n'ai pas essayé :
– Portmap
– Lpd (avec un peu d'imagination)
– Gated
– Routed
– Identd
– Comsat
– Talkd

– Tftpd
– Sadmind

La plupart des autres services peuvent probablement être tout simplement arrêtés.

3.7 Renseigner le syslog

Le programme syslog qui accompagne les systèmes à base de System V exploite /dev/log et peut utiliser d'autres *devices* du style `/dev/*log`. C'est très pratique car il peut continuer à servir à vous avertir. Il vous suffit de recréer ces *devices* dans l'environnement chrooté. J'ai essayé sur Solaris et ça fonctionne. Sur des systèmes non basés sur System V comme Linux, il faudra probablement envoyer des paquets syslog vers la machine elle-même.

Partie IV : Pour aller plus loin

4.1 Comment savoir si vous vous trouvez dans un environnement chrooté

Certains ont voulu savoir s'ils se trouvaient en environnement chrooté ou pas. Tous les points de montage du système de fichier UFS avaient une inode numérotée 2. Ce qui signifiait que la racine / chrootée n'était pas un point de montage, le numéro d'inode serait différent. Comme la plupart des administrateurs systèmes utilisent chroot sans s'en inquiéter, le code suivant peut vous dire si l'environnement chrooté est un point de montage ou pas.

```
#include <stdio.h> #include <stdlib.h> #include <sys/types.h> #include <sys/stat.h> int main(int argc, char **argv) {{   struct stat x;    if (stat("/", &x)) {{     printf("Unable to stat /");     exit(EXIT_FAILURE);  }}    if (x.st_ino==2) {{     printf("I am not chrooted or chrooted on a mountpoint\n");   }} else {{ printf("I am chrooted\n");   }}   exit(EXIT_SUCCESS); }}
```

Cependant si vous n'avez pas un environnement système complet (c'est-à-dire pas de fichier shadow, pas de binaire 'su' , ou autre chose tout aussi sympathique), il est facile de savoir que vous êtes dans un environnement chrooté.

Des administrateurs recréent un environnement système complet pour leurs utilisateurs afin qu'ils ne s'aperçoivent pas qu'ils travaillent dans un environnement chrooté. Si l'inode de / est 2, vous pouvez suspecter que votre environnement de travail est chrooté. Vous pouvez alors le vérifier de la façon suivante :
– Contrôlez la présence de certains fichiers comme le /etc/shadow (pour els shadow password) ou le binaire 'su'.
– Établissez la liste des processus et repérez ceux qui correspondent aux binaires que vous soupsçonnez être chrootés (mais certains systèmes d'exploitation autorisent les administrateurs à cacher les processus).
– Si l'OS a un répertoire /proc, cherchez où est /proc.
– Lancez la commande 'df' et vérifiez si d'autres systèmes de fichiers sont montés . Si c'est le cas, les voyez vous ? Sinon, cela devient suspect.

4.2 Casser une cellule chrootée

Casser une cellule chrootée bien faite peut s'avérer être
très difficile, mais avec les outils adéquats, on peut y arriver. Pour que quelqu'un puisse casser la cellule chrootée, il doit être capable de :
– Trouver une faille de sécurité dans un des logiciels tournant dans la cellule,
– Exploiter cette faille et trouver un moyen d'envoyer du code en mémoire ou sur le disque,
– Une fois chargé en mémoire, arriver à l'exécuter en tant que root,

– Une fois chargé sur le disque, envoyer aussi un shell pour exécuter ce code en tant que root.

Envoyer et exécuter le code malicieux en mémoire peut être aussi simple que d'exécuter un code en tant que root dans un shell comme cela a déjà pu être montré dans de nombreux *exploits*. Le code complet pour casser une cellule chrootée fait à peine 139 bytes ! Une fois élaboré, casser la cellule est relativement simple. Attention, le scénario qui suit n'est réalisable que si le logiciel en cours tourne en tant que root. Il est de fait fortement recommandé, pour rendre chroot plus puissant, de démarrer les applications systèmes avec des utilisateurs sans privilège. Cela rendra beaucoup plus difficile l'effraction de la cellule.

L'effraction sort du cadre de ce document, mais le lien suivant constitue un bon point de départ pour une explication :
http://www.bpfh.net/computing/docs/chroot-break.html.

Avec beaucoup de méthode et en comprenant bien les vulnérabilités de chroot, il doit être possible de rendre une prison particulièrement difficile à casser. Il existe également d'excellents outils pour intercepter les appels système (comme ceux lancés par les shells root par exemple) afin d'améliorer la sécurité de votre OS.

4.3 Les erreurs à ne pas commettre

Même dans un environnement chrooté, démarrer un logiciel avec des trous de sécurité dégrade considérablement la sécurité fournie par chroot. Ce qui suit est une liste des erreurs fréquentes que les administrateurs peuvent commettre en chrootant des processus :

– **Erreur :** Exécuter des processus chrooté avec des privilèges root.
– **Niveau critique :** Important

Une des pensées illusoires les plus courantes est de supposer qu'exécuter le logiciel en tant que root dans un environnement chrooté soit sûrs. C'est complètement faux. L'idée qui régit chroot est de protéger votre système contre l'exploitation d'une vulnérabilité du logiciel cible, par un intrus. Si de fait votre logiciel est vulnérable, le lancer en tant que root peut compromettre votre système de nombreuses façons. Bien sûr, chroot vous protège de la plupart des *script kiddie*, mais un pirate expérimenté peut facilement casser la cellule (s'il a des privilèges root), s'amuser avec les devices dans /dev chrooté , tuer, tracer, snooper des processus. Et, si il est animé de mauvaises intentions, il peut causer beaucoup de ravages depuis la cellule.

– **Solution :** Modifiez l'identifiant de l'utilisateur et du groupe après le chrootage. Cela peut être fait manuellement en changeant le code source avec les appels des fonctions setuid() et setgid(), ou en écrivant un petit emballage tout autour (on parle de *wrapper*). Comme le logiciel s'exécute dans un environnement chrooté, vous devriez être capable de changer les permissions pour les attribuer à un utilisateur autre que root. Par exemple, si vous chrootez Qpopper, vous pouvez le modifier de façon à ce qu'il s'execute en 'nobody' (sans utilisateur), mais dans le groupe ayant accès à toutes les boîtes aux lettres. Ceci vous permettra d'exécuter le service de messagerie sans être le super-utilisateur.

– **Erreur :** Utiliser une copie du fichier de mot de passe caché (shadow password)
– **Niveau critique :** Haut

Dans le cas de Qpopper, nous avions copié le fichier shadow dans le répertoire chrooté afin de permettre aux utilisateurs de s'authentifier. Malheureusement, si le programme popper en lui même n'est pas vulnérable, un intrus peut essayer de décrypter ce fichier shadow, utiliser un outil sécifique, puis avoir accès à tout le système en passant par la grande porte.

– Solution : Il en existe quelques-unes pour y remédier. D'abord, si vous savez installer et configurer une méthode externe d'authentification (par exemple RADIUS, SecurID, ou même en utilisants les librairies PAM), vous mettrez en place une authentification sans utiliser le fichier shadow à l'intérieur de la prison. Il s'agit de la la meilleure solution. Si elle n'est pas envisageable dans votre cas, un autre moyen de contourner le problème consiste à analyser le fichier shadow avant de le copier, d'en enlever tout utilisateur système ou ayant des privilèges. Si un compte privilégié doit absoluement être présent dans le fichier, utilisez des mots de passe et des adresses mails différents de ceux utilisés pour le système.

– Erreur : Créer des environnements système virtuels
en dehors d'un point de montage
– niveau critique : Léger

En chrootant un répertoire qui ne fait pas également office de point de montage, vous fournissez un petit trou de sécurité qui permettra à l'utilisateur de savoir s'il s'agit d'un environnement chrooté. Comme nous l'avons vu ci-dessus, tous les points de montage ont une inode dont le numéro est 2. En chrootant en dehors d'un point de montage, vous permettez donc de savoir que le processus est chrooté. Cette erreur n'est pas très dangereuse en soit puisqu'elle ne permet pas de casser cet environnement chrooté. Elle fournit néanmoins une information que l'intrus ne devrait pas savoir.

Si vous avez un environnement chrooté relativement restreint où les fichiers systèmes sont bien cachés, il n'y a pas de raison de changer quoique ce soit, mais si vous essayez de charger un environnement complet (c'est-à-dire une machine virtuelle), placer la prison dans un point de montage est une bonne idée.

– Solution : Si vous essayez de charger un environnement complet (une machine virtuelle) et que vous ne voulez pas faire savoir aux utilisateurs qu'ils sont chrootés, placez la prison dans un point de montage.

Post-scriptum :

La version originale de cet article est consultable en ligne : http://www.networkdweebs.com/chroot.html.

---

[1] **Note sur les bibliothèques dynamiques :** Si vous pouvez disposer des sources du programme que vous souhaitez chrooter, vous devriez le recompiler avec des bibliothèques statiques. Cela vous évitera de copier
également ces bibliothèques. Si, vous utilisez un système d'exploitation dont les bibliothèques sont fréquemment mises à jour, il est alors préférable de recompiler le programme avec les bibliothèques dynamiques.

[2] Il s'agit ici de l'auteur de l'article, Jonathan A. Zdziarski