LINUX: chroot overview

Table of Contents

chroot (change root) est un appel système qui a également donné son nom à une commande des systèmes d'exploitation Unix permettant de changer le répertoire racine d'un processus de la machine hôte.

Présentation

Cette commande permet d'isoler l'exécution d'un programme et d'éviter ainsi la compromission complète d'un système lors de l'exploitation d'une faille. Si un pirate utilise une faille présente sur l'application chrootée, il n'aura accès qu'à l'environnement isolé et non pas à l'ensemble du système d'exploitation. Cela permet donc de limiter les dégâts qu'il pourrait causer. Cet environnement est appelé un **chroot jail** en anglais, littéralement une prison.

Il permet également de faire tourner plusieurs instances d'un même ensemble de services ou démons sur la même machine hôte.

Par exemple, il est possible avec chroot d'exécuter des applications 32 bits sur un système 64 bits : il suffit pour cela d'avoir un sous-système qui intègre toutes les bibliothèques logicielles nécessaires ; chroot permet de se connecter à l'intérieur de ce sous-système et d'y exécuter les applications installées.

chroot est un outil GNU faisant partie de coreutils et plus précisément des shellutils.

Limiation de la prison

Lorsqu'on construit un système dans un chroot (par exemple avec debootstrap) et si on compare **/proc** à **rootfs/proc**, **/sys** à **rootfs/sys** et enfin **dev** à **rootfs/dev** on constate que les deux premiers sont vides dans **rootfs** alors qu'ils y a du contenu. Le dossier **dev** quant à lui est clairement moins

complet.

- **/proc** est un point de montage représentant un système de fichiers en mémoire qui contiendra des fichiers et dossiers (virtuels) représentant les processus lancés sur le système et leurs état. le programme “ps” lit tout simplement le contenu de ce dossier et le représente dans une forme “human” readable. Par exemple pour avoir des information sur le processus avec l’identifiant 1 (le script init, premier programme dans l’espace utilisateur) on peut exécuter `cat /proc/1/cmdline` qui retourne`/sbin/init splash` C’est la première commande appelée par le noyau linux.
- **/sys** est un point de montage représentant un système de fichier en mémoire qui permet au noyau de fournir depuis l’espace utilisateur des informations sur la configuration matérielle, les drivers chargés. Et de pouvoir modifier certains paramètres du noyau. Certains “fichiers” sont donc éditables. `lspci` par exemple utilise `sysfs` pour afficher l’ensemble des périphériques disponibles
- **/dev** : représente des périphériques à entrée/sorties qui se comportent comme des fichiers. A savoir que nous pourrons lire ou écrire des données au même titre qu’en lisant des fichiers. On retrouvera les disques durs/avec ses partitions, mais aussi les cartes d’acquisitions, claviers, souris. Par exemple `head -c 1k /dev/sda` affiche le premier kilo octet contenu dans le disque dur.

Maintenant on va utiliser la commande chroot qui permet de changer le répertoire racine pour un processus donné, en définissant le dossier **rootfs** comme racine:

```
chroot rootfs
```

Si à présent le dossier rootfs est la racine nous on constate qu'on ne peut plus monter au dessus, donc si on exécute un processus grâce a chroot il ne devrait pas pouvoir modifier des fichiers qui sont à l’extérieur de ce nouveau dossier racine. On a donc une isolation au niveau des fichiers.

Lorsqu'on observe le contenu des dossier **/proc**, **/sys** ils sont tous deux vides. Si on essaye de lancer la commande `ps` qui permet de confirmer qu’elle observe bien le contenu de **/proc**.

```
ps -ax
```

On constate que le programme demande d’exécuter cette commande

```
Error, do this: mount -t proc proc /proc
```

Il faut donc de monter le dossier **/proc** comme demandé et réexécuter la commande ps.

```
mount -t proc proc /proc
ls /proc
ps -ax
```

Le programme ps affiche tous les processus existants et le dossier **/proc** est plein.

On constate cependant qu'on a eu accès en tant que root à l’ensemble des processus de la machine. Cependant lorsqu'on tente d’arrêter un processus avec la commande `kill IDPROCESSUS`, on

constate qu'on peux arrêter tous les processus que l'on souhaite.

On constate la même chose pour le dossier **/sys**,il faut exécuter la commande suivante pour l'initialiser :

```
mount -t sysfs sys /sys
```

On a le même contenu et on peut donc lister tous les périphériques mais aussi interagir avec le noyau.

Enfin pour le dossier **/dev** le contenu ne se monte pas directement dans le chroot il faut faire un lien depuis système de fichier racine avec le dossier **/dev** racine afin d'accéder aux disques durs, caméras etc.. À l'extérieur du chroot il faudrait exécuter :

```
sudo mount -o bind /dev rootfs/dev
```

En conclusion on a pu créer un nouvel environnement grâce à Chroot qui protège le système de fichier racine mais qui ne permet pas de protèger les processus ni de limiter la communication avec le noyau pour l'administrer et gérer les ressources via sysfs.

Création d'une Debian chrooté

Pour créer un système Debian on utilisera **deboostrap** qui permet de télécharger les paquets nécessaire pour créer une distribution basée sur Debian (donc Ubuntu et autres ...).

```
sudo apt-get install debootstrap
```

Debootstrap

La distribition GNU / Linux Debian dispose d'un outil permettant de créer très simplement des prisons chroot. Il s'agit de **debootstrap**.

Commande de base

Le principe est assez simple: Le programme **debootstrap** installe dans un répertoire une arborescence complète d'une Debian minimale (Sans noyau et sans environnement graphique).

Debootstrap va permettre de créer un système de fichier complet.

On choisit l'architecture, la branche, tout le reste est automatique. La commande:

```
debootstrap [OPTION...] SUITE TARGET [MIRROR [SCRIPT]]
```

Par exemple pour une Debian jessie minimal amd64:

```
debootstrap --arch=amd64 jessie chroot http://httpredir.debian.org/debian/
```

- **Architecture:** `--arch=i386` amd64, arm64, armel, armhf, i386, mips, mipsel, powerpc, ppc64el, ou s390x
- **Branche**: wheezie, jessie, testing, ...

  Une fois terminé, on a une arborescence complète dans le répertoire cible:

```
cd /var/local
ls custom
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run
sbin  srv  sys  tmp  usr  var
```

Options

Une option intéressante permet de pré-installer certains paquets: –include linux-image-amd64,grub-pc,locales

```
cd /var/local
debootstrap --include linux-image-amd64,grub-pc,locales --arch=amd64 jessie
custom http://httpredir.debian.org/debian/
```

L'option `--print-debs` qui va lister les packages à télécharger/installer

```
cd /var/local
debootstrap --print-debs --arch=amd64 jessie custom
http://httpredir.debian.org/debian/

  I: Retrieving InRelease
  I: Retrieving Release
  I: Retrieving Release.gpg
  I: Checking Release signature
  gpgv: Signature made Sat Jan 14 14:04:51 2017 EAT
  gpgv:                using RSA key 8B48AD6246925553
  gpgv: Good signature from "Debian Archive Automatic Signing Key
(7.0/wheezy)  <ftpmaster@debian.org>"
  gpgv: Signature made Sat Jan 14 14:04:51 2017 EAT
  gpgv:                using RSA key 7638D0442B90D010
  gpgv: Good signature from "Debian Archive Automatic Signing Key (8/jessie)
<ftpmaster@debian.org>"
  gpgv: Signature made Sat Jan 14 14:32:44 2017 EAT
  gpgv:                using RSA key CBF8D6FD518E17E1
  gpgv: Good signature from "Jessie Stable Release Key <debian-
release@lists.debian.org>"
```

```
  I: Valid Release signature (key id
75DDC3C4A499F1A18CB5F3C8CBF8D6FD518E17E1)
  I: Retrieving Packages
  I: Validating Packages
  I: Resolving dependencies of required packages...
  I: Resolving dependencies of base packages...
  I: Found additional required dependencies: acl adduser dmsetup insserv
libaudit-common libaudit1 libbz2-1.0 libcap2 libcap2-bin libcryptsetup4
libdb5.3 libdebconfclient0 libdevmapper1.02.1 libgcrypt20 libgpg-error0
libkmod2 libncursesw5 libprocps3 libsemanage-common libsemanage1 libslang2
libsystemd0 libudev1 libustr-1.0-1 procps systemd systemd-sysv udev
 I: Found additional base dependencies: libdns-export100 libffi6 libgmp10
libgnutls-deb0-28 libgnutls-openssl27 libhogweed2 libicu52 libidn11 libirs-
export91 libisc-export95 libisccfg-export90 libmnl0 libnetfilter-acct1
libnettle4 libnfnetlink0 libp11-kit0 libpsl0 libtasn1-6
acl adduser base-files base-passwd bash bsdutils coreutils dash debconf
debconf-i18n debianutils diffutils dmsetup dpkg e2fslibs e2fsprogs findutils
gcc-4.8-base gcc-4.9-base grep gzip hostname init initscripts insserv
libacl1 libattr1 libaudit-common libaudit1 libblkid1 libbz2-1.0 libc-bin
libc6 libcap2 libcap2-bin libcomerr2 libcryptsetup4 libdb5.3
libdebconfclient0 libdevmapper1.02.1 libgcc1 libgcrypt20 libgpg-error0
libkmod2 liblocale-gettext-perl liblzma5 libmount1 libncurses5 libncursesw5
libpam-modules libpam-modules-bin libpam-runtime libpam0g libpcre3
libprocps3 libselinux1 libsemanage-common libsemanage1 libsepol1 libslang2
libsmartcols1 libss2 libsystemd0 libtext-charwidth-perl libtext-iconv-perl
libtext-wrapi18n-perl libtinfo5 libudev1 libustr-1.0-1 libuuid1 login lsb-
base mawk mount multiarch-support ncurses-base ncurses-bin passwd perl-base
procps sed sensible-utils startpar systemd systemd-sysv sysv-rc sysvinit-
utils tar tzdata udev util-linux zlib1g apt apt-utils bsdmainutils cpio cron
debian-archive-keyring dmidecode gnupg gpgv groff-base ifupdown init-system-
helpers iproute2 iptables iputils-ping isc-dhcp-client isc-dhcp-common kmod
less libapt-inst1.5 libapt-pkg4.12 libboost-iostreams1.55.0 libdns-export100
libestr0 libffi6 libgdbm3 libgmp10 libgnutls-deb0-28 libgnutls-openssl27
libhogweed2 libicu52 libidn11 libirs-export91 libisc-export95 libisccfg-
export90 libjson-c2 liblogging-stdlog0 liblognorm1 libmnl0 libnetfilter-
acct1 libnettle4 libnewt0.52 libnfnetlink0 libp11-kit0 libpipeline1 libpopt0
libpsl0 libreadline6 libsigc++-2.0-0c2a libssl1.0.0 libstdc++6 libtasn1-6
libusb-0.1-4 libxtables10 logrotate man-db manpages nano net-tools netbase
netcat-traditional nfacct readline-common rsyslog tasksel tasksel-data
traceroute vim-common vim-tiny wget whiptail
  I: Deleting target directory
```

Afin d'économiser votre bande passante et du temps :

- Si on dispose d'un cache local de paquet (apt-cacher-ng par exemple) on peut remplacer MIRROR par http://IP:PORT/httpredir.debian.org/debian/.
- Si on dispose d'un live CD Debian récent, remplacer MIRROR par file:/cdrom/debian/ (ou /cdrom/debian est le point de montage du CD...

Avec l'option –variant=minbase tu aura le minimum de chez minimum...

```
    --variant=minbase|buildd|fakechroot|scratchbox
```

Les variantes de script d'amorçage à utiliser, actuellement prises en charge sont les suivantes:

- **minbase**, qui ne comprend que les paquets essentiels et apt;
- **buildd**, qui installe les paquetages essentiels à la construction dans TARGET;
- **fakechroot**, qui installe les paquets sans privilèges root.
- **scratchbox**, qui permet de créer des cibles pour l'utilisation de scratchbox.

La valeur par défaut, lorsqu'on n'indique pas l'argument `--variant=X`, consiste à créer une installation Debian de base dans TARGET.

Création d'un système de fichier racine basé sur une Debian Jessie minimal.

Avec la commande ci-dessous on va créer un système Debian dans le dossier rootfs. `–variant=minbase` permet d'installer le minimum de paquet nécessaire pour que Débian fonctionne.

```
sudo debootstrap --variant=minbase jessie rootfs
```

Lorsque l'opération est terminée on obtien dans le dossier `rootfs` un système Debian prêt à l'emploi.

En effet en faisant :

```
cd rootfs && ls
```

On constate que l'architecture des dossiers ressemble fortement au dossier racine. Toutefois on observant le dossier plus en détails on constate que les certains dossiers sont vides alors que sur le système hôte ils ne le sont pas.

Création d'un CentOS chrooté

Il n'existe rien de semblable au paquet debbotstrap pour les distributions basées sur RPM aussi un travail manuel est inévitable.

Préparation de l'environnement

Tout d'abord, il faut que des programmes rpm et yum soient installés. Ensuite, on peut initialiser la base de données RPM:

```
$ mkdir -p /storage/schroot/centos-6.5
$ rpm --root=/storage/schroot/centos-6.5 --rebuilddb
```

Cela ajoutera une base de données vide à `/storage/schroot/centos-6.5/var/lib/rpm`. Pour que yum fonctionne, il faut installer quelques fichiers de configuration. Le moyen le plus simple de les obtenir pour une distribution consiste à installer le paquetage centos-release.

```
wget
http://mirror.centos.org/centos/6.5/os/x86_64/Packages/centos-release-6-5.el
6.centos.11.1.x86_64.rpm
  <skip details>

rpm --root=/storage/schroot/centos-6.5 --nodeps -i centos-
release-6-5.el6.centos.11.1.x86_64.rpm
  warning: Generating 12 missing index(es), please wait...
  warning: centos-release-6-5.el6.centos.11.1.x86_64.rpm: Header V3 RSA/SHA1
Signature, key ID c105b9de: NOKEY

rm centos-release-6-5.el6.centos.11.1.x86_64.rpm
```

Cela installera les fichiers de configuration du référentiel dans ${chroot}/etc/yum.repos.d/. Modifier ces fichiers en fonction des besoins: je souhaite personnellement que certains référentiels supplémentaires soient activés dans CentOS-Base.repo. De plus, yum veut vérifier la signature du paquet RPM, voici donc quelques options à prendre en compte:

Mieux vaut désactiver la vérification de la signature (temporairement) dans le fichier de configuration CentOS-Base.repo.

On peut maintenant passer à l'étape suivante - installer un système de base minimal:

```
yum --installroot=/storage/schroot/centos-6.5 update
```

```
base                                                     | 3.7 kB  00:00:00
centosplus                                               | 3.4 kB  00:00:00
contrib                                                  | 2.9 kB  00:00:00
extras                                                   | 3.4 kB  00:00:00
updates                                                  | 3.4 kB  00:00:00
(1/5): contrib/primary_db                                | 1.2 kB  00:00:00
(2/5): base/primary_db                                   | 4.4 MB  00:00:01
(3/5): centosplus/primary_db                             | 1.5 MB  00:00:01
(4/5): extras/primary_db                                 |  19 kB  00:00:03
(5/5): updates/primary_db                                | 3.2 MB  00:00:04
Resolving Dependencies
--> Running transaction check
---> Package centos-release.x86_64 0:6-5.el6.centos.11.1 will be updated
```

```
---> Package centos-release.x86_64 0:6-5.el6.centos.11.2 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package              Arch     Version              Repository    Size
================================================================================
Updating:
 centos-release       x86_64   6-5.el6.centos.11.2  updates       20 k

Transaction Summary
================================================================================
Upgrade  1 Package

Total download size: 20 k
Is this ok [y/N]: y
Downloading packages:
warning: /storage/schroot/centos-6.5/var/cache/yum/updates/packages/centos-
release-6-5.el6.centos.11.2.x86_64.rpm: Header V3 RSA/SHA1 Signature, key ID
c105b9de: NOKEY
Public key for centos-release-6-5.el6.centos.11.2.x86_64.rpm is not
installed
centos-release-6-5.el6.centos.11.2.x86_64.rpm
|  20 kB  00:00:00
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
Importing GPG key 0xC105B9DE:
 Userid     : "CentOS-6 Key (CentOS 6 Official Signing Key) <centos-6-
key@centos.org>"
 Fingerprint: c1da c52d 1664 e8a4 386d ba43 0946 fca2 c105 b9de
 Package    : centos-release-6-5.el6.centos.11.1.x86_64 (installed)
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
Is this ok [y/N]: y
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating   : centos-release-6-5.el6.centos.11.2.x86_64            1/2
  Cleanup    : centos-release-6-5.el6.centos.11.1.x86_64            2/2
  Verifying  : centos-release-6-5.el6.centos.11.2.x86_64            1/2
  Verifying  : centos-release-6-5.el6.centos.11.1.x86_64            2/2

Updated:
  centos-release.x86_64 0:6-5.el6.centos.11.2

Complete!
```

```
yum --installroot=/storage/schroot/centos-6.5 install -y yum
```

Reconfiguration de l'environnement chrooté

Afin d''installer le reste à partir de l’environnement chrooté, il faut nettoyer la base de données RPM.

chroot centos-6.5 bash-4.1# rm -rf /var/lib/rpm bash-4.1# rpm –rebuilddb bash-4.1# rpm –nodeps -i /var/cache/yum/updates/packages/centos-release-6-5.el6.centos.11.2.x86*64.rpm warning: /var/cache/yum/updates/packages/centos-release-6-5.el6.centos.11.2.x86*64.rpm: Header V3 RSA/SHA1 Signature, key ID c105b9de: NOKEY warning: /etc/yum.repos.d/CentOS-Base.repo saved as /etc/yum.repos.d/CentOS-Base.repo.rpmorig bash-4.1# yum install yum <skip details>

```
On peut ensuite utiliser yum groupinstall sur les distributions basées sur RHEL comme suit:
```

yum -y groupinstall “Base” –installroot=/your/path/ –releasever=6 ```