Les Jails en FreeBSD: Concepts et mise en oeuvre

Félix-Antoine Bourbonnais

fbourbonnais@rubico.info

18 janvier 2006

Groupe LinuQ,
Québec

Première partie

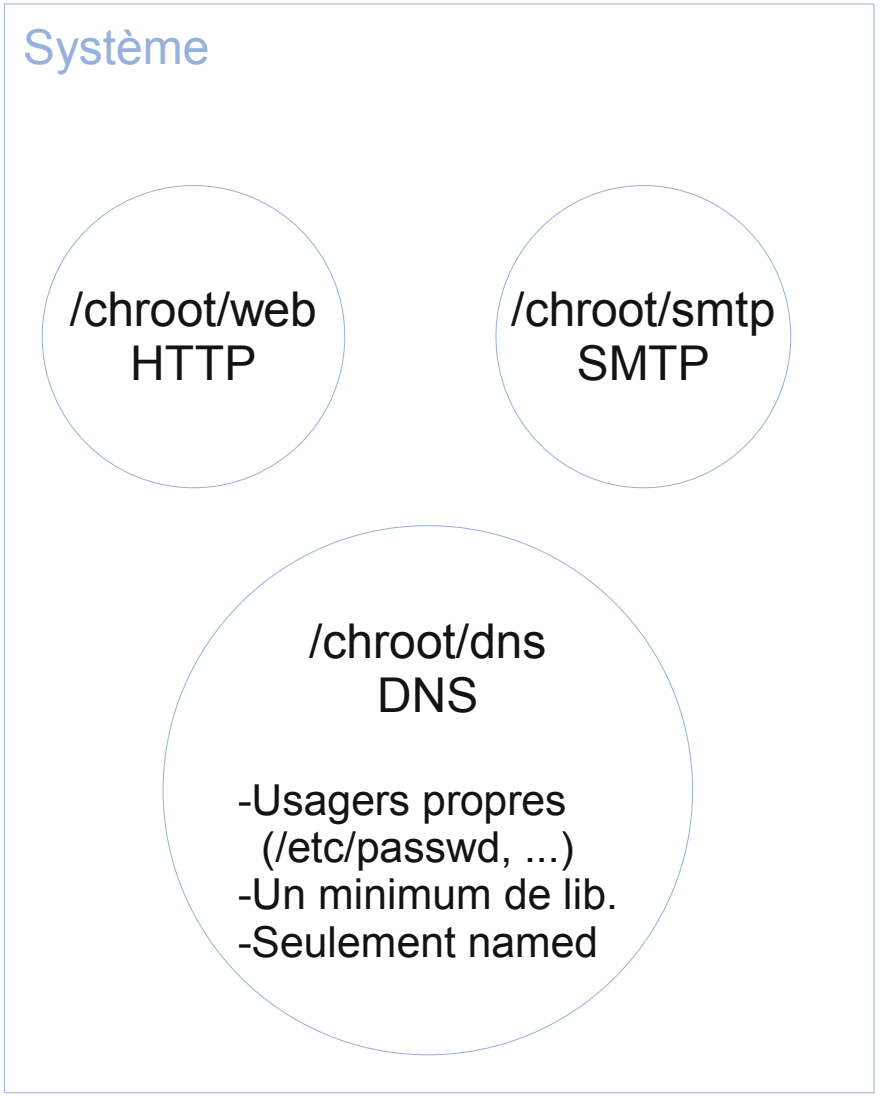
•Introduction

√ Le ***grand dictionnaire terminologique*** (GDT) de l'Office québécois de la langue française est utilisé à •titre de référence ainsi que pour les traductions linguistiques. [www.granddictionnaire.com]

Diviser pour sécuriser

- Problèmes:
  - Certains processus tournent avec des privilèges élevés (root)
  - Certains services peuvent présenter des failles de sécurité
  - En cas d'attaque, un processus compromis pourrait:
    - Trouver d'autres services à attaquer
    - Attaquer un autre processus
    - Lire, effacer ou altérer des données d'un autre service
    - Espionner d'autres services ou le réseau
  - Des logiciels peuvent nécessiter des environnements différents (libraires, configurations, etc).
    - Exemple: librairies x86_64 vs i386

Diviser pour sécuriser



- Idée: diviser pour sécuriser
- Cloisonner les services afin d'augmenter la sécurité
- Limiter les dégâts si un processus est compromis
- Les actions d'un processus ont une portée limitée
- Environnements indépendants
- On simule plusieurs systèmes indépendants à l'intérieur d'un seul système.

Les solutions

- Solutions possibles:
  - Chroot
    - Limité au système de fichier
    - Failles connues
  - Jails (FreeBSD)
    - Uniquement logiciel
  - Machines virtuelles (Xen, VMWare, ...)
    - Perte de performance
  - Plusieurs ordinateurs en réseau
    - Plus sécuritaire
    - Nettement plus coûteuse

Utilisations possibles

- Catégories d'utilisateurs:
  - [Le prévoyant] (sécurité)
    - Isoler des services autonomes
    - Exemple: Emprisonner Apache, Postfix, Bind, ...
  - [Le fournisseur internet] (délégation)
    - Offrir des accès à des systèmes complets indépendants à plusieurs utilisateurs
    - Exemple: Espaces d'hébergement pour plusieurs utilisateurs
  - [Le développeur] (virtualisation)
    - Avoir plusieurs systèmes de tests avec des env. différents
    - Exemple: Plusieurs systèmes de dev. : x86 vs x86_64

Deuxième partie

- Les chroot

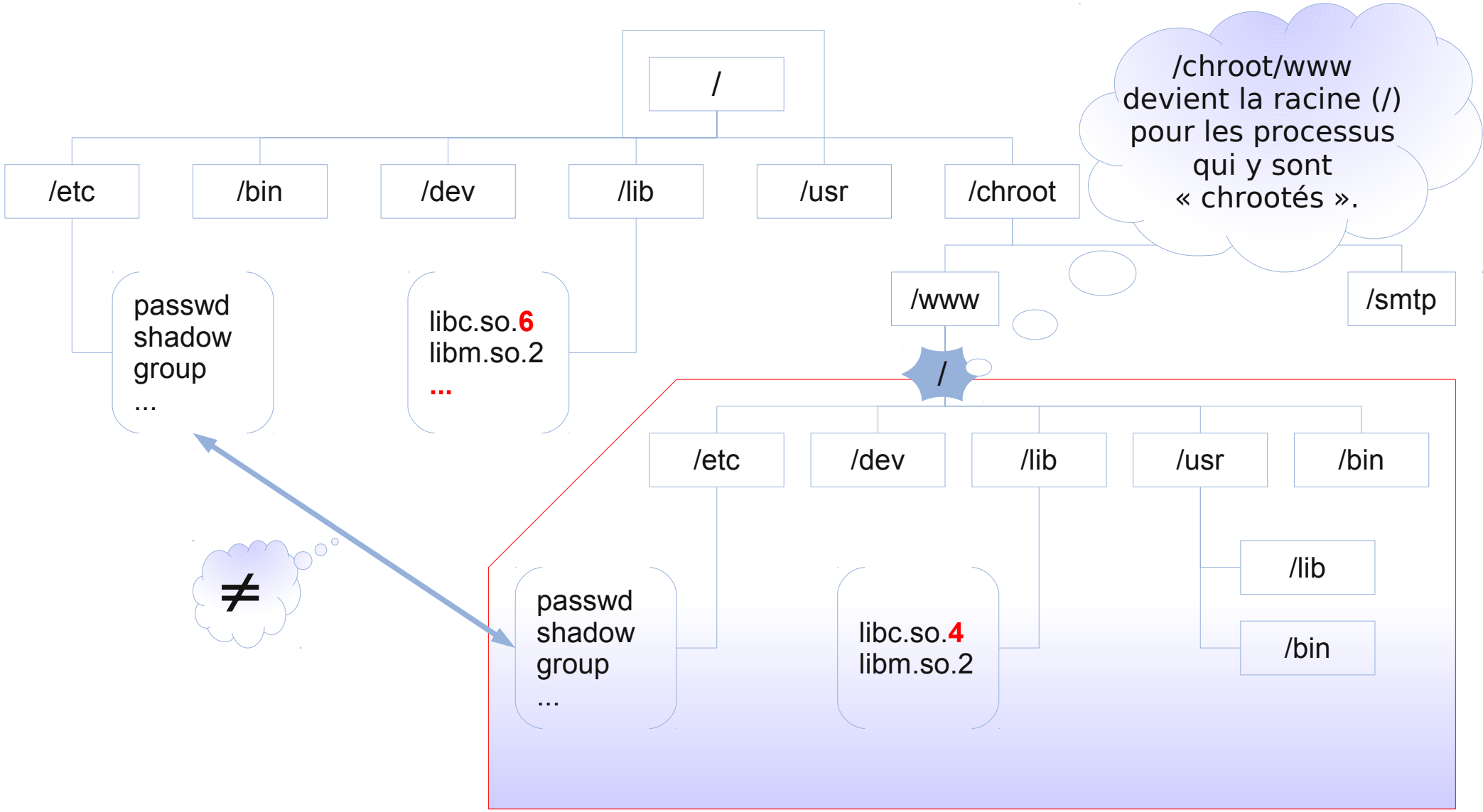
Les chroot

- Permet de déplacer la racine du système de fichiers pour un processus
- Le processus ne peut accéder aux fichiers situés plus haut dans l'arborescence.
- Permet de limiter la vision du système de fichiers pour un service
- Possède une hiérarchie indépendante
  - Utilisateurs propres à chaque chroot
  - Librairies
  - Configurations
  - Utilisateurs
- Pas de limites sur les IO, réseau et autre ressources

∝∃ La sous-arborescence est parfois appelée:

- ∝ Sandbox
- ∝ Jail (à ne pas confondre avec les jails sous FreeBSD!)

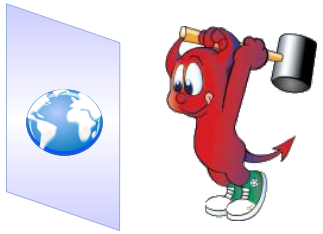
Les chroot



Troisième partie

- Les jails de FreeBSD

Les jails de FreeBSD



- ~ ATTENTION ~
  - Les chroot sont parfois appelés jails.
  - Il ne s'agit alors pas de jails tels que disponibles sous FreeBSD
  - Les jails tel que présentés ici n'existent pas sous GNU/Linux.

- Isole les processus au point de vue:
  - Système de fichiers
    - Comme les chroot
    - Les problèmes de sécurité sont corrigés
  - Réseau
    - Une seule IP
    - Impossible de modifier la configuration réseau (adresses, interface, routage, ...)
  - Système
    - Restrictions concernant les « system-calls »
    - Impossible de charger des modules
    - Monter/démonter des périphériques
    - Créer des « devices nodes » (mknod)
    - ...

Le système de fichiers

- Fonctionne exactement comme les chroot
- Les problèmes de sécurité liés au chroot ont été corrigés
- L'accès aux périphériques (/dev) est contrôlé par des règles d'accès (devfs_ruleset) [FBSD>5]
- Il est possible d'empêcher la modification de drapeaux (flags) sur les fichiers (security.jail.chflags_allowed) [FBSD>6]

Le réseau

- Chaque Jail a une adresse IP propre
  - Toute tentative d'écouter sur toutes les adresses (0.0.0.0/INADDR_ANY) sera ignorée et remplacée par l'IP du Jail
  - Comment ajouter des adresses IP à une interface ?
    - ifconfig xl0 alias 192.168.0.50 netmask 255.255.255.255
- Il est possible de prévenir la modification du « hostname » à partir du jail (jail_set_hostname_allow)
- Il est possible d'empêcher l'utilisation d'autres protocoles que TCP/IP (jail_socket_unixiproute_only)

Réseau (2)

- Impossible d'accéder à certains types de socket:
  - « Raw » (avec en-tête)
  - « Divert » (interception/filtrage)
  - « Routing »
- Ne fonctionne pas avec IPv6
- Dans le système principal: ne pas écouter sur toutes les adresses afin d'éviter des « conflits » entre le système principal et les Jails.
    - Exemple:

      Sys.: 192.168.0.1/24; SSH (22/tcp sur 0.0.0.0)

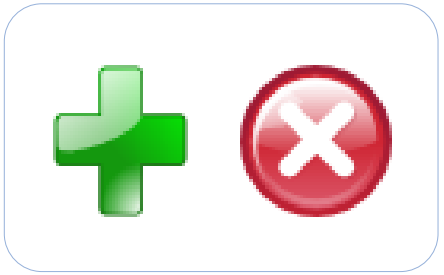
      Jail1: 192.168.0.2/32; SSH (22/tcp sur 192.168.0.2)

      Les deux écoutent sur le port 22/tcp de 192.168.0.2!

Le système

- Natif sur FreeBSD
  - C'est un « system call »
  - Uniquement logiciel. On s'intéresse à l'interaction avec le noyau.
- L'utilisateur root (d'un jail) n'est pas Dieu
- Ne peut voir que les processus qui roulent dans le même Jail
- Possibilité de désactiver les « system calls » de bas niveau (jail_sysvipc_allow)
- Impossible de changer le « security level »
- La majorité des paramètres systèmes/noyau sont bloqués (sysctl)
- Impossible de charger des modules

Avantages et inconvénients

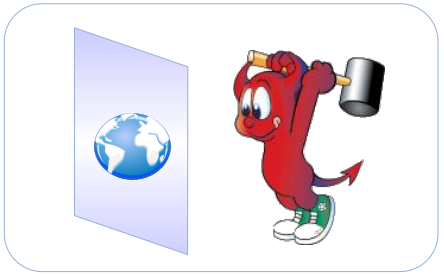


- Avantages
  - Plus sécuritaire que les chroot
  - Plus rapide que la machine virtuelle (ex.: Xen)
  - Configuration et gestion relativement simples (FBSD > 5)
  - Stabilité
- Désavantages
  - Disponible uniquement sous FreeBSD
  - Reste souvent moins sécuritaire que la séparation physique
  - Ne fonctionne pas avec IPv6
  - Rend plus complexe la maintenance et la gestion du système (sécurité vs simplicité)

Savoir dire non aux Jails

- Quand ne PAS utiliser des Jails:
  - Demande un accès direct au réseau
    - Exemple: DHCP
  - Nécessite des « system calls » de bas niveau (permettre jail_sysvipc_allow)
    - Permettre « sysvipc » revient à donner accès à la mémoire du système principal... Mauvaise idée !
    - Exemple: PostgreSQL
  - Quand la sécurité n'est pas importante
    - Chroot sera alors le candidat idéal

Quand utiliser des Jails ?



- Besoin de sécurité
- L'isolement physique (plusieurs machines) est trop coûteux ou impossible
- Nécessité de donner des accès élevés à plusieurs personnes mais qui ne doivent pas avoir accès à tout le système
  - Exemple: FAI
  - Exemple: Des étudiants doivent apprendre à administrer Apache.
- Avoir plusieurs environnements de développement ou de tests
- Un service a besoin d'un environnement complètement différent des autres

Et SELinux ??

- Les Jails vs SELinux
  - Moins sécuritaire que SELinux
  - Moins flexible que SELinux
  - Beaucoup plus simple à configurer et mettre en place que SELinux
    - L'augmentation de la complexité s'avère parfois être l'ennemi de la sécurité!

Quatrième partie

Mise en oeuvre

Les configurations

- Éditer rc.conf (FBSD>5)

```
ifconfig_xl0_alias0="inet 192.168.0.2 netmask 255.255.255.255"
syslogd_flags="-ss -l /jails/www/var/run/log"

jail_enable="YES"
jail_list="www"
jail_set_hostname_allow="NO"
jail_socket_unixiproute_only="NO" #TCP/IP Uniquement
jail_sysvipc_allow="NO"

jail_www_rootdir="/jails/www"
jail_www_hostname="www.home.examples.com"
jail_www_ip="192.168.0.2"
jail_www_fdescfs_enable="NO"
jail_www_procfs_enable="NO"
jail_www_devfs_enable="YES"
jail_www_devfs_ruleset="devfsrules_jail"
```

Configurations (2)

- Configurations de Sysctl possibles (voir man jail)
  - security.jail.allow_raw_sockets
  - security.jail.enforce_statfs
  - security.jail.set_hostname_allowed
  - security.jail.socket_unixiproute_only
  - security.jail.sysvipc_allowed
  - security.jail.chflags_allowed
- Ajouter un alias à une interface
  - ifconfig xl0 alias 192.168.0.2 netmask 255.255.255.255
  - Valide uniquement jusqu'au prochain démarrage

Configurations (3)

- Démarrer un jail (FBSD>5)
  - Manuellement
    - Un Shell

      ```
      jail /jails/www www.home.examples.com 192.168.0.2
      /bin/sh
      ```

    - Amorçage complet

      ```
      jail /jails/www www.home.examples.com 192.168.0.2
      /etc/rc
      ```

  - Automatiquement
    - Tous les Jails

      ```
      /etc/rc.d/jail start
      ```

    - Le jail www (www doit être dans « jail_list » dans « rc.conf »)

      ```
      /etc/rc.d/jail start www
      ```

Comment faire ?

- La documentation est souvent orientée vers des Jails complets (un système complet)
- J'ai expérimenté plusieurs techniques qui conviennent à des besoins différents
- Techniques:
  - Système complet (standard)
  - Jails from scratch
  - Par élimination (ma préférée)
  - ...
- Gestion des Jails
  - Création
  - Mise à jour

Par système complet

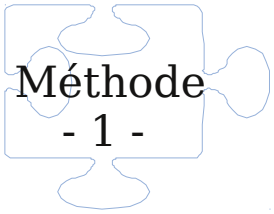
- Création :

```
D=/here/is/the/jail
cd /usr/src
mkdir -p $D
make world DESTDIR=$D
make distribution DESTDIR=$D
ifconfig xl0 alias 192.168.0.2 netmask 255.255.255.255
mount_devfs devfs $D/dev
```

- Mise à jour:
  - La mise à jour est effectuée dans le Jail dans la mesure où chaque Jail a son propre /usr/ports et /usr/src

Par système complet (2)

- C'est la méthode « standard »
- Avantages:
  - Simplicité de la création
  - Simplicité de la maintenance
- Désavantage
  - Longue compilation
  - Demande beaucoup d'espace disque pour chaque Jail
  - Tout y est... y compris les programmes dangereux!



Jails from scratch

- Création:
  - Partir de rien
  - Créer manuellement l'arborescence
  - Copier uniquement les fichiers nécessaires (conf, lib, bin, ...)
  - Tester... Essayer... Tester... Encore essayer..... BINGO!
- Astuce: Utiliser ldd pour voir toutes les librairies utilisées par un binaire ou une librairie

```
# ldd /bin/ls

/bin/ls:
libutil.so.5 => /lib/libutil.so.5 (0x28079000)
libncurses.so.6 => /lib/libncurses.so.6 (0x28085000)
libc.so.6 => /lib/libc.so.6 (0x280c4000)
```

Jails from scratch (2)

- Astuce: Utiliser « truss » pour observer les appels au système afin de repérer les fichiers que le programme essaie d'ouvrir.

```
truss ls /tmp/p
[...]
open("/etc/libmap.conf",0x0,0666)   ERR#2 'No such file or
directory'
open("/var/run/ld-elf.so.hints",0x0,00)          = 3 (0x3)
read(0x3,0xbfbfea10,0x80)                        = 128 (0x80)
lseek(3,0x80,SEEK_SET)                           = 128 (0x80)
read(0x3,0x28076000,0x3c)                        = 60 (0x3c)
close(3)                                         = 0 (0x0)
access("/lib/libutil.so.5",0)                    = 0 (0x0)
open("/lib/libutil.so.5",0x0,00)                 = 3 (0x3)
fstat(3,0xbfbfea50)                              = 0 (0x0)
[...]
```

Jails from scratch (3)

- Avantages
  - Les Jails sont de petites tailles
  - Il n'y a que le nécessaire
  - Certains diront que c'est plus sécuritaire... sujet à discussion!
- Désavantages
  - Peut s'avérer long à construire
  - Risque d'erreurs élevé

• Par élimination

- Variante de la méthode 1
- Consiste à enlever tout le superflu après la génération du jail
- Méthode personnelle et inspirée de Nate Nielsen[1]
- Automatiser le processus avec mon script de création
  - Mon script est disponible sur demande (écrivez-moi)

1 Nate Nielsen, http://memberwebs.com/nielsen/

Par élimination (2)

- Il est possible d'empêcher la compilation de certains composants lors du makeworld via le fichier /etc/make.conf
  - Faire deux fichiers /etc/make-std.conf et /etc/make-jails.conf
  - Enlever le maximum de composants superflus dans make-jails.conf. (ex.: NO_CXX=, NO_MAN=, ...)
  - Créer un lien symbolique /etc/make.conf vers /etc/make-std.conf pour compiler le système principal et vers /etc/make-jails.conf pour compiler un jail.
- Il n'est pas nécessaire de recompiler tout le code entre chaque création.
  - Si le contenu de /usr/src n'a pas changé depuis le dernier « buildworld »
  - Si make.conf n'a pas changé
  - Si /usr/obj n'a pas été effacé
  - Alors, effectuer uniquement un « installworld »!

Par élimination (3)

- Script pour créer le Jail
  - 0- Fabriquer des paquetages pour les ports (pkg_create -b)
  - 1- Création (make installworld)
  - 2- Élimination du superflux
  - 3- Installation des paquetages via chroot
- Solution hybride pour la mise à jour
  - Ports: « pkg_create -b » ou « make package »
  - Sources: copies ou « installworld » par chroot+mount, ...

Annexe I

À propos

Remerciements

Remerciements

Frédéric Lebel

Pour son aide lors de la préparation de cette présentation

Présentation en ligne

Cette présentation est disponible en ligne:

http://www.rubico.info/docs/jails-freebsd/

Annexe II

Références

Références

FreeBSD Jail Software and Docs
Nate Nielsen

http://memberwebs.com/nielsen/freebsd/jails/

Secion6Wiki, Creating a FreeBSD Jail

http://www.section6.net/wiki/index.php/Creating_a_FreeBSD_Jail

(attention, il y a des erreurs dans les scripts!)

FreeBSD man Jail(8)

http://www.freebsd.org/cgi/man.cgi?query=jail&apropos=0&sektion=0&manpath=FreeBSD+6.0-RELEASE&format=html

Références

- *Jails: Confining the omnipotent root*, FreeBSD Project
  http://docs.freebsd.org/44doc/papers/jail/jail.html

- How to break out of a chroot() jail
  http://www.bpfh.net/simes/computing/chroot-break.html

- [Attaques possibles sur un Jail (dans le bas)]:
  http://lists.nycbug.org/pipermail/talk/2005-May/005569.html

Références

- *Managing Jails on FreeBSD 5*:
  http://www.devco.net/archives/2005/02/13/managing_jails_on_freebsd_5.php
- *BIND in chroot jail*:
  http://lists.freebsd.org/pipermail/freebsd-questions/2004-January/032841.html
- *Postfix in a FreeBSD jail*:
  http://barnson.org/node/139
- *Using jail to imprison processes and their descendants for increased security*:
  http://www.ezunix.org/modules.php?
  op=modload&name=Sections&file=index&req=viewarticle&artid=48&page=1

Les Jails en FreeBSD

- *Using a jail as a virtual machine (FBSD 4.X)*:

  http://www.freebsddiary.org/jail.php

- *FreeBSD Jail scripts* (FBSD 4.X):

  http://jailnotes.cg.nu/zcripts/