# 1 Création d' un environnement sécurisé pour la suite Mozilla

## 1.1 introduction

La Fondation Mozilla sort régulièrement de nouvelles versions pour ces logiciels phares, firefox et thunderbird. Sous Linux, les notions élémentaires de sécurité et de cohérence du système nous amène à utiliser la version rpm. Il n' est pas question d' avoir un répertoire type heu.. program files ;) sur lequel n' importe qui aurait tout les droits et dans lequel on mettrai un gros bordel parfois redondant (même en renseignant un fichier pour qu' un nouveau programme s' y trouvant sache si une lib qui lui est nécéssaire est déjà présente ou pas...) Tout ceci serait non-sens et je ne rentrerai pas dans plus de détails ici sur le sujet.

Mozilla-firefox est pensé, prévu, pour dodoze, c' est clair, c' est limpide. Il embarque notament une fonction interne de mise-à-jour. Celle-ci n' est pas utilisable pour Linux, car il faudrait lancer firefox avec des droits root pour qu' il est droit d' écriture dans les répertoires nécessaires. Lancer un navigateur internet en root !! autant se jeter du 7ème étage de suite... Non, on est sous le nunux et l' on aime bien :) Donc essayons de conserver sa logique sécuritaire tout en oeuvrant pour avoir un firefox qui permette de se mettre à jour tout seul.

pré-recquis :
** connaissance & notions de l' architecture du système*
** connaissance & notions de sécurité*
** connaissance d' un minimum de shell, bash & sh*
** connaissance de la compilation de sources*
** ne pas avoir peur d' apprendre*

on va voir :
** les notions d' un shell restreint*
** l' enchrootage du shell*
** la compilation spécifique à la Suite de la Fondation Mozilla*

niveau :
** je vous laisse juges*

objectifs :
** Proposer un environnement sécurisé à Mozilla-Firefox*
** Conserver l' ensemble de ses fonctions*
** Avoir un ensemble cohérent permettant de séparer tout codes et fichiers temporaires du /home courant*
** Avoir bien plus et bien mieux que le "container IE" que proposera Vista (pour un simple équivalent, voir le forum ou les précédents articles de ce blog)*
**Avoir une pérénité du système : une fois mis en place, on utilisera simplement la fonction de mise à jour interne à firefox*

## 1.1 Création d' une nouvelle partition

Pourquoi une nouvelle partition ? Ceci n' est pas une étape obligatoire
Pour cloisonner encore plus.

## 1.1 Création d' un nouvel utilisateur du système

On ouvre un émulateur de terminal, puis on se loggue en root. On y lance drakuser qui va nous permettre de facilement réussir cette première étape. (une note au passage, sur le forum était indiqué que l' on pouvait entrer directement n' importe quel chemin comme shell dans drakuser. Ceux connaissant Redhat savent que ce n' est pas possible avec leur outil de gestion des utilisateurs, pourtant très proche graphiquement. sur RedHat/Fedora, on doit se contenter du choix prédéfinis entre les divers shell installés, et basta. Ceux voulant le changer vers un répertoire, un fichier, non conventionnel iront le faire dans les fichiersde conf et à la main... Et hop, encore un petit plus pour Mandriva-Linux)

Pour l' exemple on va nommer ce nouvel utilisateur par le doux pseudonyme de "mozillasuite". Par convention, le terme mozillasuite fera désormais référence à cet utilisateur. Les programmes seront quant à eux simplement appelés par leurs noms dans la suite de l' article.

{image:http://club.mandriva.com/xwiki/bin/download/bubar/Securite3/capture18.jpg}

On peut voir ici deux choses : Tout d' abord le mot de passe, il n' est pas très long, ceci pour faciliter pour l' instant une manipulation rapide des diverses connections sous ce compte que nous devrons faire. A la fin de l' opération, nous changerons sont mot de passe pour quelque chose de plus sérieux, d' une longueur de plus de 14 caractères avec des minuscules, des majuscules et des chiffres. Un mot de passe normal en somme. Deuxièmement, le shell : nous laissons pour le moment bash.

{image:http://club.mandriva.com/xwiki/bin/download/bubar/Securite3/capture19.jpg}

On ajoute cet utilisateur au groupe audio, pour faciliter l' utilisation du multimédia puisque un des objectifs est d' avoir un firefox parfaitement fonctionnel pour l' ensemble des ses possibilités. On verra plus tard certaines finesses dans le contexte.

## 1.1 Notion : Le shell restreint

Bash vient par défaut dans un gros rpm, bash veut dire Bourne Again SHell, je ne vais pas faire l' historique du bash ici, mais intéressons plus particulièrement à l' option -r lors de l' invocation de bash. Cette option permet d' obtenir out-of-the-box un bash restreint dont voici la liste des fonctions qui, pour un utilisateur placé sous ce bash, seront impossible à effectuer :

*Changer de répertoire avec la commande "cd".
*Changer la valeur des variables d'environnement : $SHELL, $PATH, $ENV et $BASH_ENV.
*Executer une commande contenant au moins un slash (/).
*Passer en argument un fichier contenant au moins un slash (/) aux commandes "." et "source".
*Passer en argument un fichier contenant au moins un slash (/) à la commande "hash ?p".
*Lecture et traitement des valeurs de SHELLOPTS au démarrage du shell.
*Importer des définitions de fonctions au démarrage du shell.
*Utiliser les opérateurs de redirection de flux ('>', '', '>&', '&>' et '>>').
*Utiliser la commande "exec".
*Ajouter ou supprimer des commandes avec les options "?f" et "?d" de "enable".
*Utiliser la commande "command" avec l'option ?p.
*Annuler le mode restrictif avec les commandes 'set +r' or 'set +o restricted'.

On le voit, c' est rempli de choses intéressantes. Ce n' est pas le but de cet article de traiter du restricted-bash et je ne vais pas détailler toutes les finesses, qui ne sont pas utiles à connaitre ici. Ce bash amputé d' outils à des fins de restrictions est assez usité dans le cadre de connections ssh, un admi n' est jamais assez parano avec ces users... Ce bash peut être invoqué par la commande rbash ou encore bash --restricted ou plus simplement -r

On peut deviner à quoi il va nous être utile ici... Mais continuer sur ce chemin...

## 1.1 Notion : Le chroot

Le chroot est un mot valise pour désigner change-root. "On change le root". C' est à dire que l' on va créer un environnement fonctionnel, taillé à la serpe juste pour nos besoins spécifiques. celui ci sera inclus dans l' environnement courant. Cet environnement sera fonctionnel dans les strictes limites qu' on lui donne et ne pourra pas interagir avec le système réel. Une commande chroot est disponible dans /usr/sbin/. Cette commande est également usitée pour créer une distribution chrootée et se déplacer dedans en y executant des commandes de cette distribution. (personnellement je fais tourner 3 linux en même sur le même ordinateur sans émulation aucune, en en chrootant 2 dans le "réel") Ce n' est pas sujet ici et je ne m' étendrais pas plus sur les fonctionnalités et les limites du chrootage. Nous ne ferons pas ici un chroot complet, nous nous contenterons du strict nécessaire. L' utilisation usuelle de cette commande est : chroot /PATH_DU_NOUVEAU_ROOT /COMMANDE_A_CHROOTER

## 1.1 Chrooter bash pour mozillasuite, et lui donner rbash

Revenons à notre bash classique et occupons de la chrooter, d' en enfermer une copie dans le /home de mozillasuite, nous restreindrons ainsi grandement les interactions avec le système lui même. Là nous allons attaqué de la ligne de commande et un peu de code :

* créer les répertoires nécessaires
* copier le binaire nécessaire, bash
* vérifier les librairies dépendantes à bash
* préparer un shell bidon qui en fait enchrootera bash
* indiquer ce chell bidon comme shell pour mozillasuite
* modifier le lancement du bash pour cet utilisateur

```
[mozillasuite@athing ~]$ pwd
/home/mozillasuite
[mozillasuite@athing ~] mkdir ./lib ./bin ./sbin ./etc ./usr ./dev
[mozillasuite@athing ~] exit

[root@athing ~]# rm -rf /homemozillasuite/.bash*
[root@athing ~]# chown mozillasuite:mozillasuite /home/mozillasuite/bin/
[root@athing ~]# chown mozillasuite:mozillasuite /home/mozillasuite/lib/
```

... faites de même pour chacun des répertoires crées ci-dessus

Créons maintenant le device null :)

```
[root@athing ~]# mknod /home/mozillasuite/dev/null c 1 3 -m 666
```

Copions le binaire bash, le shell initial, dans 'l' arborescence système secondaire' du /home de mozillasuite.

```
[root@athing ~]# cp /bin/bash /home/mozillasuite/bin/
```

Mais bash est dépendant de nombreuses librairies, il nous faut donc les dénicher et les copier à l' identique dans son /lib

Pour les dénicher, on utilise la commande ldd :

```
[root@athing ~]# ldd /bin/bash
        linux-gate.so.1 =>  (0xbfffe000)
        libtermcap.so.2 => /lib/libtermcap.so.2 (0xb7ef2000)
        libdl.so.2 => /lib/libdl.so.2 (0xb7eef000)
        libc.so.6 => /lib/i686/libc.so.6 (0xb7dc1000)
        /lib/ld-linux.so.2 (0xb7f12000)
[root@athing ~]#
```

ATTENTION vérifiez bien les librairies sur VOTRE système avec la commande ldd puis recopier les librairies désignées par le résultat uniquement chez VOUS.
Nous voilà renseigner, il faut donc copier ces librairies dans le futur chroot, 'l' arborescence système secondaire'. sans se préoccuper de la première, la linux-gate.so.1

```
[root@athing ~]# cp /lib/libncurses.so.5 /home/mozillasuite/lib/
[root@athing ~]# cp  /lib/libtermcap.so.2 /home/mozillasuite/lib/
[root@athing ~]# cp  /lib/libdl.so.2 /home/mozillasuite/lib/
[root@athing ~]# cp /lib/ld-linux.so.2 /home/mozillasuite/lib/
[root@athing ~]# cp /lib/libc.so.6 /home/mozillasuite/lib
```

Voilà, le chroot est prêt à être utiliser, il nous reste à créer le shell bidon qui indiquera quoi faire lors de la connection. allons y :

```
[root@athing ~]# touch /bin/logtoi
```

```
#!/bin/bash
exec -c /usr/sbin/chroot /home/$USER /bin/bash -r
```

remarquez l' option -r pour --restricted (ou Rbash) qiu invoque bash dans un mode restreint. Remarquez également l' option -c à exec, qui permet d' attacher le processus directement à init (le père de tout les process) sans que bien sûr mozillasuite ai un quelconque droit dessus, cela permet donc de détacher ce process de tout autre et de l' attacher à init.

Maintenant, on va modifier le path du shell de démarrage pour ce user, mozillasuite. Ceci se fait en éditant le fichier /etc/passwd (que, rappel, nous pouvons modifier facilement avec drakuser...). Dans ce fichier on trouve à la fin de chaque ligne les renseignements concernant le /home et le shell. C' est donc ce dernier qui nous intéresse, et l' on va remplacer bien sûr l' actuel par /bin/logtoi

## 1.1 Sécurisons un peu tout cela

En premier lieu, on doit ajuster les droits de l' ensemble des fichiers dans le /home de mozillasuite. Également de donner le droit de lecture et d' entrée à tous sur son /home :

```
[root@athing ~]# chown -R mozillasuite:mozillasuite /home/mozillasuite/
[root@athing ~]# chmod -R 600 /home/mozillasuite/
[root@athing ~]# chmod +rx /home/mozillasuite/
```

En second lieu, il convient de créer les fichiers passwd et shadow et d' y ajuster les droits pour root exclusif dessus, afin d' éviter que mozillasuite ne puisse sortir de son chroot facilement.

```
[root@athing ~]# touch /home/mozillasuite/etc/shadow
[root@athing ~]# touch /home/mozillasuite/etc/passwd
[root@athing ~]# chown root:root /home/mozillasuite/etc/passwd
[root@athing ~]# chown root:root /home/mozillasuite/etc/shadow
[root@athing ~]# chmod 600 /home/mozillasuite/etc/passwd
[root@athing ~]# chmod 600 /home/mozillasuite/etc/shadow
```

remarquez qu' on les laisse vide : peu importe ;) l' essentiel est que mozillasuite ne puisse pas les effacer, ni les créer ni les éditer...

Ca y est, le SHELL pour l' utilisateur mozillasuite est prêt, le système chrooté également. si maintenant on se loggue sous cette identité, nous serons chrooté dans /home/mozillafirefox avec un shell restreint.

{image:http://club.mandriva.com/xwiki/bin/download/bubar/Securite3/capture20.jpg}

Cette image nous montre que l' on passe sous l' identité mozillasuite -> cela fonctionne :) Et nous montre à quel point le shell est restrictif : tout d' abord par l' option -r on a appelé le rbash ... et de plus, /bin de cet environnement étant quasiment vide, il ne sait quasiment rien faire pour le moment. La commande pwd nous confirme qu' il se croit bien à la racine.

Notez que lorsqu' on remplir un peu plus ce shell, nous serons à l' abri de bêtises par le fait d' avoir invoqué bash en restricted... On pourrait par exemple copier exec, il ne saurait pas s' en servir... ;)

## 1.1 Téléchargement des logiciels & préparation

*zou, téléchargeons les dernières versions des logiciels firefox & thunderbird sur le site officiel de la Fondation Mozilla. Pour cela le plus facilement possible, pourquoi ne pas nous servir directement de ce nouvel utilisateur ? Nous allons ainsi nous habituer à l' utilisation de ce chroot. Il nous fait ajouter wget à l' environnement de chroot, pour se faire nous procédons tout simplement comme prédécement :*
*[root@athing ~]# cp /usr/bin/wget /home/mozillasuite/bin*
*[root@athing ~]# ldd /usr/bin/wget*
*-> on copie les librairies sorties en résultat de cette commande dans /home/mozillasuite/lib puis on oublie pas de ré-ajuster les droits :*
*[root@athing ~]# chown -R mozillasuite:mozillasuite /home/mozillasuite/lib*

*On ouvre donc un émulateur de terminal dans lequel on passe sous l' identité de mozillasuite... et on lance le rapatriement des tarball :*
*Mais mais mais le shell restreint vous en empêche ?? ;) ;) :) Bon nous verons cela dans l' épisode numéro 2 qui sera plus spécifique aux notions de sécurité, à celle-ci et à d' autres... Revenons donc à nos moutons, assez de lignes de commandes pour l' instant :)*
arf arf arf c' était une petite touche d' humour... ha ce sacré shell chrooté...

-> Rapatrions donc ces tarballs le plus facilement possible, décompressons les quelque part, puis copiant les dossiers dans le /home de mozillasuite... Mais mais mais... pourquoi prendre une version dont vous connaissez le "binss" (...) -> prenons donc les sources tant qu' faire !! Compilons les, avec les bonnes options, pour obtenir un binaire statique qui s' intègrera facilement dans notre environnement chrooté de ce cher user mozillasuite...

ftp://ftp.mozilla.org/pub/mozilla.org/firefox/releases/1.5.0.4/source/firefox-1.5.0.4-source.tar.bz2

et hop hop hop on compile le bouzin ...

## 1,1 Compiler Firefox

La Fondation Mozilla fait extrèmement bien les choses : en effet un projet / des programmes de cette ampleur serait classiquement un casse tête effroyable à compiler (y compris sur gentoo ou sur *bsd). Mais contrairement aux compilations "classiques" nous allons utiliser ici un fichier de configuration spécial qui indiquera au script quoi faire et comment, je vous livre une copie du mien, en enlevant les commentaires pour une meilleure visibilité :

Je vous présente maintenant le fichier que je vous conseille d' utiliser, celui ci est taillé pour firefox uniquement, avec bien sûr une compilation pour un résultat statique demandé.

```
. $topsrcdir/browser/config/mozconfig
mk_add_options MOZ_OBJDIR=@TOPSRCDIR@/ff-opt-static
ac_add_options --enable-optimize
ac_add_options --disable-debug
ac_add_options --enable-static
ac_add_options --disable-shared
ac_add_options --disable-tests
mk_add_options MOZ_CO_PROJECT=browser
```

Ce fichier est à bien sûr placé à la racine du /home du nouvel utilisateur ayant /bin/sh comme shell et nous servant spécifiquement à la compilation de firefox, en appelant ce fichier .mozconfig

Vous reviendrez une fois celui-ci réussi, plus facilement à un similaire pour thunderbird. Les plus aventuriers se lanceront directement avec le fichier suivant :

```
. $topsrcdir/browser/config/mozconfig
. $topsrcdir/mail/config/mozconfig
. $topsrcdir/calendar/sunbird/config/mozconfig
. $topsrcdir/xulrunner/config/mozconfig
ac_add_options --with-system-zlib
ac_add_options --with-system-png
ac_add_options --with-system-jpeg
ac_add_options --enable-system-cairo
ac_add_options --enable-canvas
ac_add_options --enable-svg
ac_add_options --enable-strip
ac_add_options --disable-tests
ac_add_options --disable-installer
ac_add_options --disable-accessibility
ac_add_options --with-system-nss
ac_add_options --with-system-nspr
mk_add_options MOZ_CO_PROJECT=browser
ac_add_options --enable-static --disable-shared
ac_add_options --enable-application=browser
ac_add_options --enable-default-toolkit=gtk2|xlib|qt|cairo-gtk2|cairo-xlib
ac_add_options --enable-xft
mk_add_options MOZ_CO_PROJECT=xulrunner
ac_add_options --enable-static --disable-shared
ac_add_options --enable-application=xulrunner
ac_add_options --enable-default-toolkit=gtk2|xlib|qt|cairo-gtk2|cairo-xlib
ac_add_options --enable-xft
mk_add_options MOZ_CO_PROJECT=calendar
ac_add_options --enable-static --disable-shared
ac_add_options --enable-application=calendar
ac_add_options --enable-plaintext-editor-only
ac_add_options --enable-necko-protocols=about,http,ftp,file,res
ac_add_options --enable-storage
ac_add_options --disable-accessibility
ac_add_options --disable-activex
ac_add_options --disable-activex-scripting
ac_add_options --disable-installer
ac_add_options --disable-mathml
ac_add_options --disable-necko-disk-cache
ac_add_options --disable-oji
ac_add_options --disable-view-source
ac_add_options --disable-logging
ac_add_options --disable-plugins
ac_add_options --disable-cookies
ac_add_options --enable-default-toolkit=gtk2|xlib|qt|cairo-gtk2|cairo-xlib
ac_add_options --enable-xft
mk_add_options MOZ_CO_PROJECT=mail
ac_add_options --enable-application=mail
ac_add_options --enable-static --disable-shared
ac_add_options --enable-default-toolkit=gtk2|xlib|qt|cairo-gtk2|cairo-xlib
ac_add_options --enable-xft
```

En lisant vous aurez certainement compris que l' on va tout compiler d' un seul coup : firefox, thunderbird & sunbird. Je ne sais pas s' il y a moyen de faire "encore plus" propre que cela, par exemple en fusionnant les options redondantes pour les passer à tous d' un seul coup.. bon ça marche :)

On confirme maintenant l' utilisatiion de ce fichier de macros de configuration :

```
export MOZCONFIG=./mozconfig-firefox
```

Et on lance la compilation :

```
make -f client.mk build
```

Patience....
Vous avez tout les atouts en main pour réussir du premier coup :)
Quelques temps (heures ?) plus tard, vous trouverez les binaires statiques utilisables, que l' on transfera alors dans /home/mozillasuite/bin (en ajustant les droits après, bien sûr)

vala vala vala ... !

## 1,1 Fin

Voilà, les logiciels sont fonctionnels
Ce qui facilite la gestion des fichiers temporaires, restreints d' éventuels codes malveillants non seulement à la notion typique des *nix, cad aux /home et droits restreints, mais en plus à une partition séparée, un /home séparé, du /home de l' utilisateur courant. Le tout dans un shell enfermé et ultra-restreint. -> Pas banal, non ;) et très sécurisé.

-> tranquilles, sûrs & sereins.
-> firefox pouvant être mis à jour depuis le site de la fondation Mozilla directement.
-> il est où vista hein, il est où ?
-> ils sont où, les malwares, hein, ils sont où ?

L' auteur de cet article (moi!) recommande fortement cette méthode pour tout les ordinateurs en environnement sensibles ou contenant des documents sensibles. Tels que des ordinateurs de gendarmes, de police, de chercheurs, d' universitaires, homme politique... avocats, huissiers.. heu non, pas ces deux derniers là ! ;) :)

## 1.1 Et si nous parlions connection ?

héhéhéhé une autre fois ;)

par Yvan -aka bubar- Munoz