codequoi

Born2beroot 02 : configurer un serveur virtuel Debian

Par Mia Combeau
Dans Projets 42
11 mars 2022
25 min de lecture
Laisser un commentaire

B

Après avoir installé la machine virtuelle pour le projet 42 Born2beroot, il nous faut configurer notre nouveau serveur. Le sujet nous demande de configurer surtout des mesures de sécurité avec la mise en place de sudo, d'un pare-feu, d'une politique de mot de passe fort, et d'un script intitulé `monitoring.sh`. De plus, nous allons ici apprendre à gérer les utilisateurs ainsi que les groupes auxquels ils appartiennent.

Dans cette série d'articles, notre serveur virtuel tourne sous Debian 11.2 (bullseye) avec VirtualBox 6.1.

Born2beroot : **Installation** | **Configuration** | **Bonus** | Sujet [pdf]

Table des matières

Commandes utiles pour Born2beroot

Les commandes de base qu'il faut connaître et qui seront sans doute très utiles lors de la configuration de notre serveur Born2beroot sont :

- **`exit`** ou **`logout`** : termine la session en cours et nous renvoie à l'écran de connexion.
- **`su autre_login`** : se connecter en tant qu'un autre utilisateur. On nous demandera alors le mot de passe de l'autre utilisateur. Lancer **`su`** (***s**witch **u**ser*) sans préciser d'utilisateur nous permettra de se connecter en tant que l'utilisateur par defaut, root. On peut **`exit`** pour retourner à la session précédente.
- **`systemctl reboot`** : redémarre le système (permissions root requises)
- **`systemctl poweroff`** : éteint le système (permissions root requises)

Gestion des paquets dans Debian : Apt et aptitude

Dans les systèmes d'exploitation de type Linux, comme Debian, l'installation de logiciels se fait à l'aide de paquets. Ces paquets contiennent tous les fichiers nécessaires à la mise en place d'un ensemble de commandes ou de fonctionnalités. Il va donc falloir comprendre comment installer ces paquets dans notre serveur virtuel Born2beroot.

APT

APT (***A**dvanced **P**ackaging **T**ool*) est un outil qui récupère les paquets auprès de sources officielles pour les installer, les mettre a jour et les supprimer ainsi que leur dépendances. On ne peut l'utiliser que par ligne de commande. Crée à l'origine pour Debian, APT vient maintenant préinstallé dans la majorité des distributions Linux.

APT est en réalité une collection de plusieurs outils, notamment : **`apt-get`**, **`apt-cache`** et **`apt-config`**. Pour installer un paquet, on peut écrire `apt install`

`[package_name]`, ce qui est équivalent à `apt-get install [package_name]`. De même, pour voir les détails d'un paquet, on peut faire `apt search [package_name]` ou `apt-cache search [package_name]`.

Aptitude

Aptitude est un outil très similaire avec le même objectif, mais c'est un gestionnaire de paquet de plus haut niveau. Il intègre les fonctionnalités d'APT, et offre une interface graphique en plus de l'interface textuelle. Cette dernière est plus informative que celle d'APT : par exemple, en annotant quels paquets sont actuellement installés ou non lors d'une recherche. Aptitude est aussi plus robuste pour la résolution de conflits lors d'installations et de suppression et on peut lui demander pourquoi tel ou tel package est recommandé avec les commandes `why` et `why-not`.

Pour installer Aptitude, il suffit de lancer la commande **`apt install aptitude`**. Pour le reste de cet article, on va continuer à utiliser APT, mais il y a des commandes équivalentes pour Aptitude.

Mise à jour des paquets

Avant d'installer un paquet, il est recommandé de toujours mettre à jour la liste des paquets du gestionnaire pour recevoir la version la plus récente. Il est aussi recommandé de mettre à jour les paquets par la même occasion. Ceci nécessite les permissions de root :

```
apt update    // met à jour la liste des paque
apt upgrade   // met à jour les paquets
```

Nb : si la mise à jour échoue avec un message comme quoi le dépôt n'est pas encore valide, c'est peut être parce que la machine virtuelle a été restaurée depuis une image et son horloge est décalée. Vérifions la date et l'heure du système avec la commande **`timedatectl`**, il se pourrait qu'on soit dans le passé ! Si c'est le cas, il suffit de redémarrer le système et attendre une petite minute pour que l'horloge se mette à jour.

Root, superutilisateur et sudo

Un simple utilisateur de notre serveur Born2beroot ne peut pas installer de paquets, ni même les mettre à jour. Pour cela, il nous faut accéder au super-utilisateur, le root. On l'a configuré lors de l'installation et avec un peu de chance on se souvient de son mot de passe !

Root, c'est véritablement l'utilisateur le plus puissant, celui à la racine d'une installation Debian. Il peut absolument tout faire sur la machine. Et, comme on dit, « à grand pouvoir grande responsabilité ». Il est bien préférable de ne jamais se connecter en tant que root où une petite erreur peut avoir de grandes répercussions. Mais on a tout de même besoin de ses privilèges de temps en temps...

sudo (***s**uper-**u**ser **do***, ou « super-utilisateur, fais ... ») est un programme qui permet à certains utilisateurs d'exécuter certaines commandes en tant que root, depuis leur propre session. Comme cela, les utilisateurs n'ont même pas besoin de connaitre le mot de passe root : `sudo` demande leur mot de passe à eux. Chaque commande `sudo` est aussi enregistrée par mesure de sécurité.

Installer sudo dans Born2beroot

Pour installer sudo, il va falloir exceptionnellement se connecter en tant que root.

```
$ su root
```

On nous demande le mot de passe root. Une fois connectés, on notera que le signe de l'invite de commande $ s'est transformé en # pour dénoter qu'on a maintenant les accès du root.

```
# apt update
# apt upgrade
# apt install sudo
# sudo --version
```

Maintenant, il faut donner à notre utilisateur normal la permission d'utiliser sudo en l'ajoutant au groupe sudo (et on va immédiatement vérifier que l'utilisateur a bien été ajouté):

```
# sudo usermod -aG sudo [mon_login]
# getent group sudo
```

Pour vérifier qu'on a bien maintenant les privilèges sudo, on peut `exit` la session root pour retourner à notre propre session, puis faire la commande :

```
$ sudo whoami
```

Après avoir saisi notre mot de passe, on devrait voir comme réponse, « root ». Si cela ne marche pas, il faudra sans doute se déconnecter et se reconnecter pour que le changement prenne effet.

En tout dernier recours, si l'utilisateur n'a toujours pas les privilèges sudo après une déconnexion/reconnexion, il faudra modifier le fichier `sudoers.tmp` depuis le root avec les commandes suivantes :

```
$ su
# sudo visudo
```

Et ajouter cette ligne au fichier :

```
mon_login   ALL=(ALL:ALL) ALL
```

Configurer sudo pour Born2beroot

Pour des questions de sécurité, le sujet de Born2beroot nous demande aussi quelques autres configurations. On doit:

- limiter l'authentification pour sudo à 3 essais en cas de mot de passe erroné,
- définir un message en cas d'erreur suite à un mauvais mot de passe,
- archiver les actions de sudo dans un journal dans `/var/log/sudo/`,
- et activer le mode TTY pour empêcher qu'un programme malveillant puisse se donner les privilèges de root via sudo.

Pour ce faire, on va ajouter les lignes suivantes au fichier `sudoers.tmp` (comme vu ci-dessus, il faut l'ouvrir en tant que root avec la commande **`sudo visudo`** pour pouvoir le modifier) :

```
Defaults     passwd_tries=3
Defaults     badpass_message="Mauvais mot de p
Defaults     logfile="/var/log/sudo/sudo.log"
Defaults     log_input
Defaults     log_output
Defaults     requiretty
```

Si le dossier `/var/log/sudo` n'existe pas, il faudra **`mkdir sudo`** dans `/var/log/`.

Maintenant, on a les privilèges de root de façon sécurisée, sans devoir se connecter en tant que root.

AppArmor pour Born2beroot Debian

AppArmor est un logiciel de sécurité pour Linux qui permet à l'administrateur système de restreindre les accès d'un programme au système d'exploitation. A chaque programme est associé un profil qui contrôle ses capacités à accéder au réseau, ses permissions de lecture, d'écriture, d'exécution, entre autres.

AppArmor est considéré une alternative à **SELinux**, un logiciel similaire mais bien plus compliqué à installer et à entretenir. Les deux programmes ont un fonctionnement diamétralement opposé. AppArmor a tendance à tout autoriser, puis à restreindre graduellement, et SELinux commence par tout refuser, puis se desserre peu à peu. Mais ce n'est pas la seule difference de fonctionnement : SELinux applique des étiquettes aux fichiers, tandis qu'AppArmor surveille plutôt les chemins d'accès.

Malgré ses difficultés d'utilisation, SELinux est probablement beaucoup plus sécurisé qu'AppArmor. On pourrait l'installer sur notre machine Debian, mais AppArmor y est déjà par défaut et il est explicitement dit dans le sujet de Born2beroot que « AppArmor pour Debian devra [...] rester actif. »

Pour vérifier qu'AppArmor est bien installé sur notre système, demandons à voir son statut :

```
$ sudo aa-status
```

Si AppArmor est là et fonctionne correctement, on devait voir quelque chose comme :

```
apparmor module is loaded.
3 profiles are loaded.
...
```

UFW pour Born2beroot

UFW (***Uncomplicated Firewall***) est, comme son nom l'indique, un pare-feu simple sur ligne de commande. Un pare-feu est un logiciel qui surveille et contrôle le trafic de données entre l'ordinateur local et le réseau. Il décide s'il bloque ou autorise le trafic selon un ensemble défini de règles de sécurité.

Adresses IP et ports

Quand on parle de trafic et de réseau, il faut comprendre deux choses, les adresses IP et les ports. Sur internet, les données sont transférées d'un ordinateur à l'autre en utilisant leurs adresses IP, qui ressemble à quelque chose comme ceci : 109.234.160.5.

Mais pour éviter les conflits entre les divers protocoles d'internet, tout ordinateur sépare les chemins d'accès à l'aide de ports spécifiés après l'adresse IP, comme ceci : 109.234.160.5:80 (port 80). Les données d'un site internet transféré via HTTP utilise le port 80. HTTPS utilise le port 443, SSH le port 22, SMTP (email sortant) le port 25, IMAP (email entrant) le port 143, etc.

Configurer UFW pour Born2beroot

Avec UFW, on va pouvoir définir des règles qui gouvernent les autorisations et les interdictions port par port. Installons et lançons UFW :

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install ufw
$ sudo ufw enable
```

Si tout se passe bien, on devrait recevoir un message comme quoi le pare-feu est actif et se lance au démarrage du système. On peut, à n'importe quel moment, vérifier le statut d'UFW et celui des ports avec la commande :

```
$ sudo ufw status verbose
```

On remarquera que, par défaut, la règle pour le trafic sortant (*outgoing*) est autorisée. Si on interdit cette regle par default avec la commande ci-dessous, le gestionnaire de paquets et autres programmes essentiels cessent de fonctionner correctement :

```
$ sudo ufw default deny outgoing
$ sudo apt update
$ sudo ufw default allow outgoing
```

La mise à jour d'APT renvoie des erreurs car elle ne peut pas utiliser son port de sortie pour chercher la liste des paquets. Il faut donc remettre la règle de sortie par défaut à autoriser.

Pour l'instant on n'a pas de règles spécifiques à un port dans notre liste lorsqu'on regarde le statut d'UFW. On peut autoriser ou interdire le port 4242 par exemple avec les commandes suivantes :

```
$ sudo ufw allow 4242
$ sudo ufw deny 4242
```

Supprimer une règle UFW

Pour supprimer une règle, comme on sera amenés à le faire lors de l'évaluation du projet, on a deux options. La première est une commande très simple. Mettons qu'on a deux règles dans notre liste qu'on veut supprimer, « allow 4242 » et « deny 4343 » :

```
$ sudo ufw delete allow 4242
$ sudo ufw delete deny 4343
```

Ces règles sont maintenant supprimées.

L'autre façon de faire c'est d'afficher la liste des règles préfixées de numéros et de supprimer la règle depuis son index. Attention, si on compte supprimer plusieurs règles de cette façon, les indexes risquent de changer après la suppression de chaque règle.

```
$ sudo ufw status numbered
$ sudo ufw delete 2
```

Et voilà, c'est tout pour UFW. Assez simple, non ?

SSH pour Born2beroot

SSH, ou **S**ecure **Sh**ell, est un protocole réseau qui permet de se connecter à distance via un réseau non sécurisé. A l'aide des mots de passe utilisateurs et la création de clefs publiques et privées, les données communiquées entre les deux ordinateurs sont cryptées.

Configurer OpenSSH pour Born2beroot

Dans Born2beroot, on nous demande d'installer ce protocole et de faire en sorte qu'il utilise le port 4242. L'outil **OpenSSH** est le plus populaire et le plus répandu,

alors installons celui-là. Vu qu'on veut pouvoir se connecter à cette machine depuis une autre, il nous faut le paquet **`openssh-server`**. Pour pouvoir se connecter à une autre machine depuis notre machine Born2beroot, il faudrait le paquet `openssh-client`, mais ce n'est pas le cas ici.

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install openssh-server
```

Lançons la commande suivante pour vérifier le statut du serveur SSH, on devrait voir « active » en vert :

```
$ sudo systemctl status ssh
```

On peut maintenant voir que le serveur SSH écoute sur le port 22. Il nous faut changer ce port au port 4242. On peut faire cela en modifiant le fichier de configuration ssh :

```
$ sudo nano /etc/ssh/sshd_config
```

Il faut trouver la ligne qu'on cherche se trouve en début de fichier et lit « #Port 22 ». On va décommenter et modifier cette ligne comme ceci : « Port 4242 » (sans le # préfixé !). Ensuite on doit relancer le service ssh pour que le changement prenne effet.

```
$ sudo systemctl restart ssh
```

Ensuite, on peut vérifier encore une fois le statut du serveur SSH, et en bas de l'affichage, on devrait voir « Server listening on :: port 4242 ».

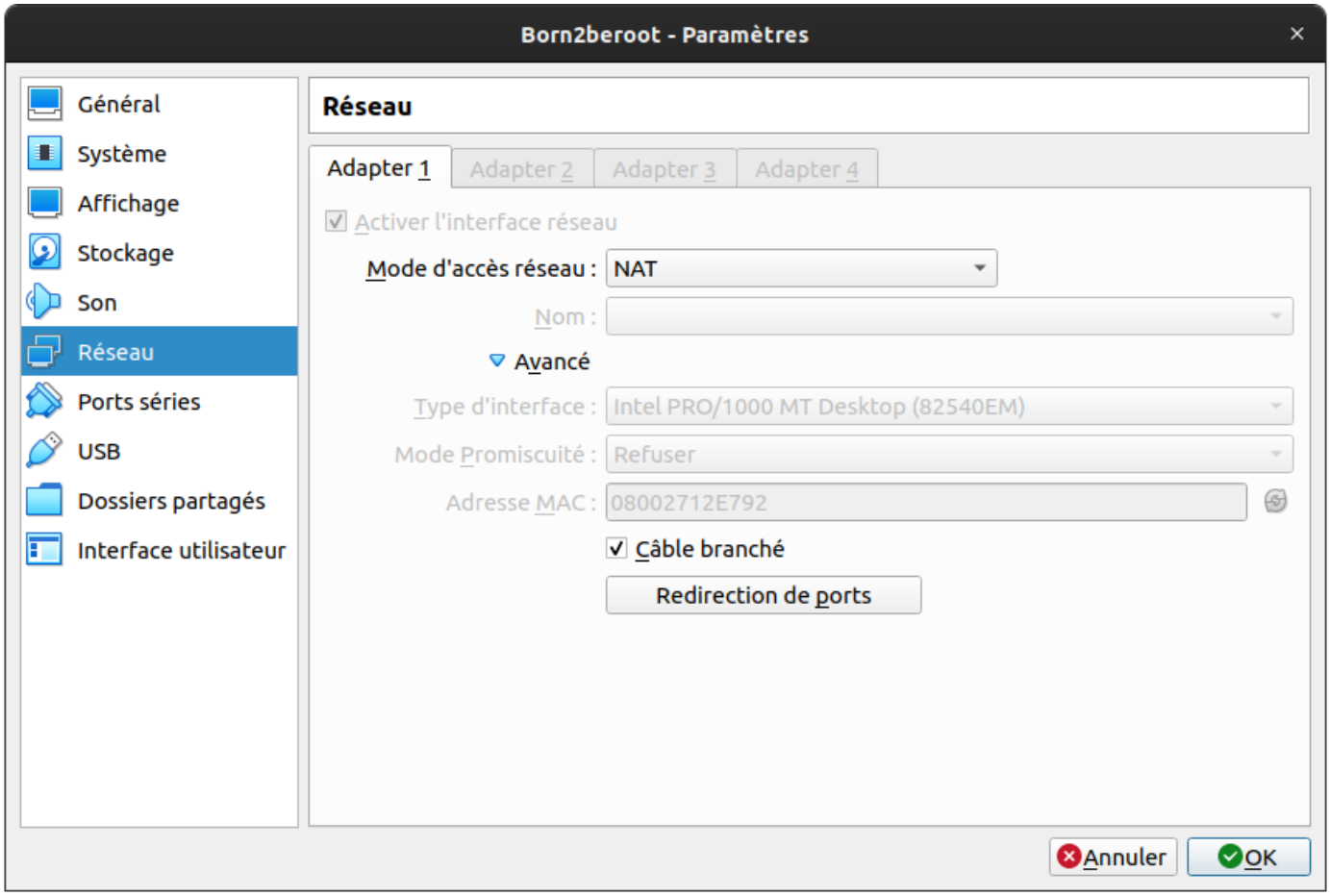
On ne doit pas oublier de dire à notre pare-feu d'autoriser la connexion au port 4242 ! Il faudra peut-être aussi supprimer une règle concernant le port 22, automatiquement ajoutée lors de l'installation d'OpenSSH.

```
$ sudo ufw allow 4242
$ sudo delete allow ssh  #Pour supprimer la rè
```

Redirection du port 4242 dans VirtualBox

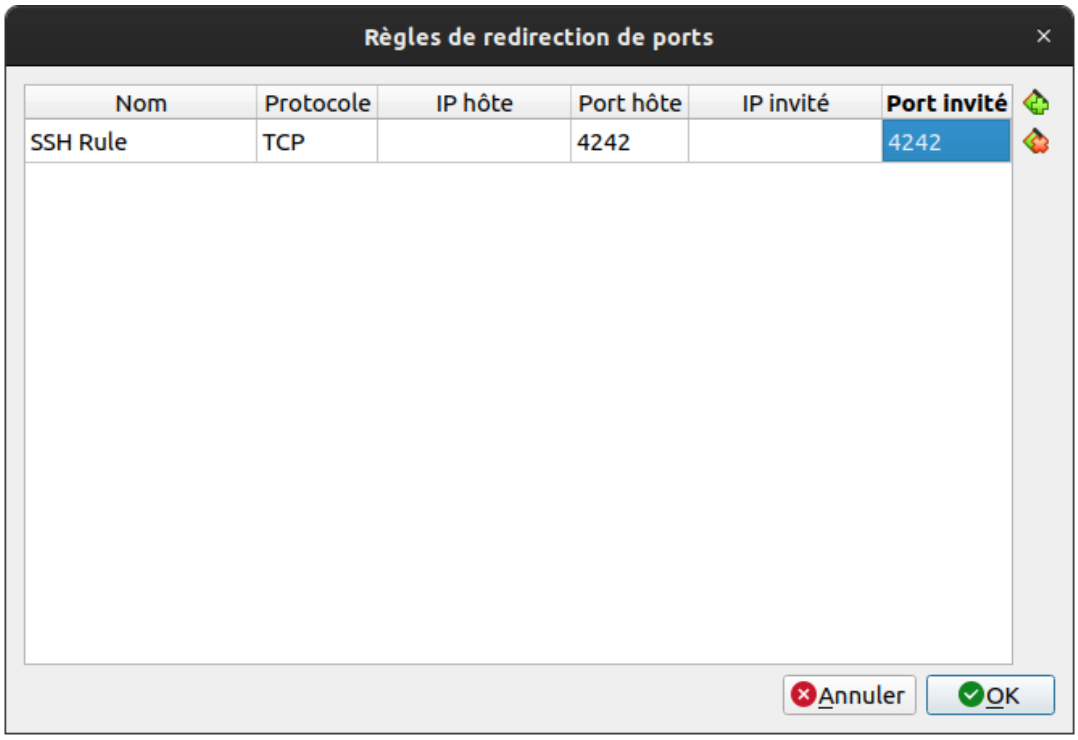
Avant de pouvoir se connecter à la machine virtuelle depuis une autre machine via SSH, il faut encore faire un petit ajustement dans VirtualBox. En effet, la connexion sera refusée à moins de rediriger le port de la machine hôte vers le port de la machine virtuelle.

Dans VirtualBox, selectionner la machine Born2beroot et aller dans configuration.



Ensuite aller dans **réseau** -> **Adapter 1** -> **Avancé** -> **Redirection de ports**. Puis on va rediriger le port hôte 4242

au port invité 4242, comme ci-dessous :



Ensuite, dans la machine virtuelle Born2beroot, on va relancer le service SSH et vérifier son statut :

```
$ sudo systemctl restart ssh
$ sudo systemctl status ssh
```

Connexion au serveur Born2beroot via SSH

Maintenant que tout est configuré, on peut vérifier la connexion via SSH en tentant de se connecter à la machine virtuelle Born2beroot depuis le terminal de notre machine hôte. Il faut évidemment que la machine virtuelle soit en marche pour s'y connecter.

Depuis le terminal d'une autre machine, on peut s'y connecter comme suit :

```
$ ssh <login_serveur>@<adresse_IP_du_serveur>
```

Le login sera évidemment celui de l'utilisateur dans la machine virtuelle et le port sera 4242. Par contre, quelle est l'adresse IP de notre serveur Born2beroot ? Vu que la machine virtuelle partage l'adresse IP de la machine hôte, on va simplement utiliser l'adresse IP du localhost. Localhost, c'est un raccourci interne que tous les ordinateurs utilisent pour indiquer leur propre adresse IP. Cette adresse est 127.0.0.1.

On peut alors transcrire la commande précédente de ces deux façons :

```
$ ssh mcombeau@localhost -p 4242
$ ssh mcombeau@127.0.0.1 -p 4242
```

Une fois le mot de passe de l'utilisateur saisi, on contrôle la machine virtuelle depuis l'extérieur ! On remarquera que notre invite de commandes aura changé et indique maintenant le *hostname* de la machine virtuelle.

Pour sortir de la connexion SSH, il suffit de taper la commande :

```
$ exit
```

Politique de mot de passe Born2beroot

Born2beroot nous demande de mettre en place une politique de mot de passe fort. Cette politique doit imposer les règles suivantes :

1. Le mot de passe devra expirer tous les 30 jours.

2. Le nombre minimum de jours avant de pouvoir modifier un mot de passe sera configuré à 2.
3. L'utilisateur devra recevoir un avertissement 7 jours avant que son mot de passe n'expire.
4. Le mot de passe sera de 10 caractères minimum
5. Il doit contenir une majuscule et un chiffre, et ne devra pas comporter plus de 3 caractères identiques consécutifs.
6. Le mot de passe ne devra pas comporter le nom de l'utilisateur.
7. La règle suivante ne s'applique pas à l'utilisateur root : le mot de passe devra comporter au moins 7 caractères qui ne sont pas présents dans l'ancien mot de passe.

Pour configurer les 3 premières règles, on doit modifier le fichier `/etc/login.defs` :

```
$ sudo nano /etc/login.defs
```

Et trouver la section « *Password aging controls* » pour en changer les valeurs tels que :

```
PASS_MAX_DAYS 30
PASS_MIN_DAYS 2
PASS_WARN_AGE 7
```

Cependant, ces changements ne s'appliqueront pas automatiquement aux utilisateurs qui existent déjà. Pour root et pour notre utilisateur, on va devoir utiliser la commande **`chage`** pour imposer ces règles. On peut utiliser le flag **`-l`** pour voir les règles appliquées à un certain utilisateur.

```
$ sudo chage -M 30 <utilisateur/root>
$ sudo chage -m 2 <utilisateur/root>
$ sudo chage -W 7 <utilisateur/root>
$ sudo chage -l <utilisateur/root>
```

Pour le reste des règles, on doit installer la librairie de vérification de qualité des mots de passe.

```
$ sudo apt install libpam-pwquality
```

Ensuite, on doit modifier le fichier `/etc/security/pwquality.conf` en supprimant la marque de mise en commentaire (#) et en changeant les valeurs associées aux options, pour avoir quelque chose comme ceci :

```
# Number of characters in the new password tha
# old password.
difok = 7
# The minimum acceptable size for the new pass
# credits are not disabled which is the defaul
minlen = 10
# The maximum credit for having digits in the
# it is the minimun number of digits in the ne
dcredit = -1
# The maximum credit for having uppercase char
# If less than 0 it is the minimun number of u
# password.
ucredit = -1
# ...
# The maximum number of allowed consecutive sa
# The check is disabled if the value is 0.
maxrepeat = 3
# ...
# Whether to check it it contains the user nam
# The check is disabled if the value is 0.
usercheck = 1
# ...
# Prompt user at most N times before returning
retry = 3
# Enforces pwquality checks on the root user p
# Enabled if the option is present.
```

```
enforce_for_root
# ...
```

Le tour est joué !

La dernière chose à faire, c'est de changer les mots de passe de l'utilisateur qui existe déjà et du root, en accord avec la nouvelle politique et avec la commande :

```
$ sudo passwd <utilisateur>
```

Hostname, utilisateurs et groupes

Pendant l'évaluation de Born2beroot, on nous demandera de changer le *hostname*, de créer un nouvel utilisateur et de manipuler les groupes d'utilisateurs. Pour cela, il va falloir connaitre les commandes ci-dessous.

Changement de hostname

Le *hostname* de la machine virtuelle doit être le **login intra + 42**. Qu'on l'ait déjà nommé correctement ou non, on sera amené à changer ce *hostname* pendant la correction. Il faut donc savoir comment. La commande ci-dessous fera le boulot :

```
$ sudo hostnamectl set-hostname <nouveau_hostn
```

On peut également modifier le *hostname* en éditant le fichier `/etc/hostname`.

Pour que le changement prenne effet, il faut redémarrer la machine, ce qui peut prendre du temps. L'alternative est

simplement de montrer le statut du hostname après le changement :

```
$ hostnamectl status
```

Modification des utilisateurs

Au démarrage, on doit avoir au moins deux utilisateurs : root et un utilisateur personnel du même nom que notre **login intra**. Pour l'évaluation, on doit aussi pouvoir montrer une liste de tous les utilisateurs, ajouter ou supprimer des utilisateurs, changer leur nom, les ajouter ou les enlever d'un groupe, etc. Les commandes pour faire tout ça sont :

- **`useradd`** : crée un nouvel utilisateur.
- **`usermod`** : modifie les paramètres de l'utilisateur : **`-l`** pour le login, **`-c`** pour le nom entier, **`-g`** pour les groupes par numéro d'identification de groupe.
- **`userdel -r`** : supprime l'utilisateur et tout ses fichiers.
- **`id -u`** : affiche le numéro d'identification de l'utilisateur.
- **`users`** : montre une liste de tous les utilisateurs connectés à l'instant.
- **`cat /etc/passwd | cut -d ":" -f 1`** : affiche la liste de tous les utilisateurs sur la machine.
- **`cat /etc/passwd | awk -F '{print $1}'`** : idem.

Manipulation des groupes

De même, on nous demandera de manipuler les groupes d'utilisateurs. L'utilisateur personnel par défaut devra être dans les groupes **sudo** et **user42**. Les commandes suivantes sont à maîtriser pour l'évaluation :

- **`groupadd`** : crée un nouveau groupe.
- **`gpasswd -a`** : ajoute un utilisateur à un groupe.

- **`gpasswd -d`** : enlève un utilisateur d'un groupe.
- **`groupdel`** : supprime un groupe.
- **`groups`** : affiche les groupes dans lesquels se trouve un utilisateur.
- **`id -g`** : montre le numéro d'identification du groupe principal d'un utilisateur.
- **`getent group`** : affiche la liste des utilisateurs dans un groupe.

Monitoring.sh pour Born2beroot

La dernière chose qu'il nous faut pour compléter la partie obligatoire de Born2beroot, c'est un petit script bash. Ce script, dans un fichier nommé `monitoring.sh`, doit afficher les informations suivantes toutes les 10 minutes dans tous les terminaux, dès le lancement de notre serveur :

- L'architecture du système d'exploitation ainsi que sa version de kernel.
  - **`uname -srvmo`**
- Le nombre de processeurs physiques.
  - **`grep 'physical id' /proc/cpuinfo | uniq | wc -l`**
- Le nombre de processeurs virtuels.
  - **`grep processor /proc/cpuinfo | uniq | wc -l`**
- La mémoire vive disponible actuelle sur le serveur ainsi que son taux d'utilisation sous forme de pourcentage.
  - Utilisé : **`free -h | grep Mem | awk '{print $3}'`**
  - Total : **`free -h | grep Mem | awk '{print $2}'`**
  - Pourcentage : **`free -k | grep Mem | awk '{printf("%.2f%%"), $3 / $2 * 100}'`**

- La mémoire disponible actuelle sur le serveur ainsi que son taux d'utilisation sous forme de pourcentage.
  - Utilisé : **`df -h --total | grep total | awk '{print $3}'`**
  - Total : **`df -h --total | grep total | awk '{print $2}'`**
  - Pourcentage : **`df -k --total | grep total | awk '{print $5}'`**
- Le taux d'utilisation actuel des processeurs sous forme de pourcentage.
  - **`top -bn1 | grep '^%Cpu' | cut -c 9- | xargs | awk '{printf("%.1f%%"), $1 + $3}'`**
- La date et l'heure du dernier redémarrage.
  - **`who -b | awk '{print($3 " " $4)}'`**
- Si LVM est actif ou pas.
  - **`if [ $(lsblk | grep lvm | wc -l) -eq 0 ]; then echo no; else echo yes; fi`**
- Le nombre de connexions actives.
  - **`grep TCP /proc/net/sockstat | awk '{print $3}'`**
- Le nombre d'utilisateurs utilisant le serveur.
  - **`who | wc -l`**
- L'adresse IPv4 du serveur, ainsi que son adresse MAC (Media Access Control).
  - IP : **`hostname -I | awk '{print $1}'`**
  - MAC : **`ip link show | grep link/ether | awk '{print $2}'`**
- Le nombre de commandes exécutées avec le programme sudo.
  - **`grep COMMAND /var/log/sudo/sudo.log | wc -l`**

Certaines des commandes ci-dessus ne marcheront pas sans les permissions de root. C'est pourquoi on va se connecter en tant que root et créer le fichier `monitoring.sh` à la racine. Il faudra aussi donner à ce fichier les droits d'exécution.

```
# chmod 755 monitoring.sh
```

Pour pouvoir exécuter le script comme on nous le demande, on doit comprendre comment faire pour diffuser un message sur tous les terminaux, et comment automatiser le script pour qu'il se déclenche toutes les 10 minutes.

La commande wall

La commande **wall** est celle qui permet d'envoyer un message à tous les utilisateurs à la fois, qui s'affiche dans tous les terminaux. Elle peut recevoir soit du texte, soit le contenu d'un fichier. Par défaut, elle préfixe l'annonce d'un bandeau. Ce bandeau est optionnel dans ce projet.

On a donc deux options :

- **`wall "message"`**
- **`wall -n "message"`** : affichage sans bandeau

Le service Cron

Cron (ou **cron**tab, ***chrono table***, qui signifie « table de planification ») est un programme qui permet d'exécuter des scripts, des commandes ou des logiciels automatiquement, à une date et heure ou un intervalle spécifié. Il est installé par défaut dans Debian (on peut vérifier avec la commande `apt list cron`). Il faut d'abord l'activer pour être certains qu'il se lance au démarrage :

```
# systemctl enable cron
```

Cron utilise des fichiers `crontab` pour planifier les tâches. Chaque utilisateur peut en avoir un ou plusieurs. En tant qu'utilisateur root, on va en créer un maintenant avec la commande :

```
# crontab -e
```

La syntaxe de cron peut sembler obscure, mais est assez facile à digérer :

```
* * * * * <commande à exécuter>
```

Les étoiles représentent des valeurs temporelles :

```
.------------------------ minute (0-59)
| .--------------------- heure (0-23)
| | .------------------ jour du mois (1-31)
| | | .------------- mois de l'année (1-12)
| | | | .--------- jour de la semaine (0-6, 0 =
| | | | |
* * * * * <commande à exécuter>
```

En remplaçant les étoiles par des valeurs numériques, on peut donc définir quand notre commande doit être exécutée.

Alors, toutes les 10 minutes, ça s'écrit comme ça, non ?

```
10 * * * * bash /root/monitoring.sh
```

Presque, mais non. L'instruction ci-dessus veut dire « exécute ceci à la dixième minute de chaque heure, tous les jours et tous les mois. » Donc notre monitoring ne s'exécuterait pas toutes les 10 minutes, mais uniquement à minuit 10, 1h10, 02h10, 3h10, 4h10, etc.

Alors comment faire? Pour dire « toutes les 10 minutes », on va « diviser » l'étoile des minutes par 10, tout simplement.

```
*/10 * * * * bash /root/monitoring.sh
```

Si l'on n'a pas incorporé la commande wall dans notre script monitoring.sh, on peut la tuyeauter dans notre règle cron comme ceci:

```
*/10 * * * * bash /root/monitoring.sh | wall
```

Cependant, le script s'exécute maintenant toutes les 10 minutes de l'heure, et non toutes les dix minutes **depuis le démarrage** de la machine, comme le demande le sujet. Pour créer le bon délai, on peut faire appel à la commande **`sleep`**, qui interrompt une commande pour un certain nombre de secondes.

Créer un délai avec sleep

Créons donc un autre script nommé **`sleep.sh`**. Dans ce script, on va calculer le nombre de secondes entre l'heure précise du démarrage de la machine et la dixième minute. Pour que ça marche, on fera appel à `bc`, la calculette en ligne de commande, qu'on peut installer avec **`apt install bc`**. Ensuite, dans `sleep.sh` :

```
#!bin/bash

# Trouver les minutes et secondes de l'heure d
BOOT_MIN=$(uptime -s | cut -d ":" -f 2)
BOOT_SEC=$(uptime -s | cut -d ":" -f 3)

# Calculer le nombre de secondes entre le déma
# Ex: si l'heure de démarrage était 11:43:36
# 43 % 10 = 3 minutes depuis la 40eme minute d
# 3 * 60 = 180 secondes depuis la 40eme minute
# 180 + 36 = 216 secondes entre le démarrage e
DELAY=$(bc <<< $BOOT_MIN%10*60+$BOOT_SEC)

# Attendre ce nombre de minutes
sleep $DELAY
```

On peut modifier de nouveau notre crontab pour qu'il prenne en compte le délai de `sleep.sh` avant d'exécuter `monitoring.sh` :

```
*/10 * * * * bash /root/sleep.sh && bash /root
```

Et voilà, notre script `monitoring.sh` s'exécutera maintenant toutes les 10 minutes **depuis le démarrage** de notre machine.

Derniers petits conseils

ERROR Failed to send host log message

Au démarrage de la machine, on remarquera peut-être l'erreur suivante : [DRM :vmw_host_log [VMWGFX]] *ERROR* Failed to send host log message. C'est une petite erreur du contrôleur graphique qui ne pose aucun problème au bon fonctionnement de la machine. C'est tout de même pas très beau à voir. On peut facilement enlever cette erreur en changeant de contrôleur graphique :

- Éteindre la machine virtuelle,
- Aller dans VirtualBox >> Machine >> **Configuration**,
- Aller dans **Affichage** >> **Écran** >> **Contrôleur graphique**,
- Choisir **VBoxVGA**.

Et voilà, il ne devrait plus y avoir d'erreur lorsqu'on redémarre la machine.

Signature.txt

Le sujet de Born2beroot explique comment extraire la signature de la machine virtuelle. Cependant, il faut savoir

que cette signature risque de changer dès qu'on modifie quoique ce soit dans la machine virtuelle. Ce qui veut dire que la signature sera correcte pour la première correction, mais aura changé avant la deuxième. Il est donc important de prendre un instantané (*snapshot*) de la machine juste avant d'extraire la signature, et de restaurer la machine depuis cet instantané entre les corrections.

C'est tout pour la partie obligatoire de Born2beroot. Il ne reste plus qu'à faire les bonus !

Bonus de Born2beroot

Les bonus de Born2beroot nous invitent à installer un site WordPress ainsi qu'un autre service de notre choix. Ils feront cependant l'objet d'un troisième et dernier article, celui-ci étant déjà bien long !

Born2beroot : **Installation** | **Configuration** | **Bonus** | Sujet [pdf]

Sources et lectures complémentaires

- Gunjit Khera, *What is APT and Aptitude? and What's real Difference Between Them?* [tecmint]
- Debian Wiki, *sudo* [Debian Wiki]
- Linuxize, *How to Add User to Sudoers in Debian* [Linuxize]
- Justin Ellingwood, Brian Boucheron, *How to Edit the Sudoers File* [DigitalOcean]
- Wikipédia, *AppArmor* [Wikipédia]
- Tuyen Pham Thanh, *AppArmor vs SELinux* [Omarine]
- TechTerms, *Port* [TechTerms]
- Port Forward, *What is a Port?* [Port Forward]

- Debian Wiki, *Uncomplicated Firewall (ufw)* [Debian Wiki]
- Debian Wiki, *SSH* [Debian Wiki]
- Bradley Mitchell, *127.0.0.1 IP Address Explained* [LifeWire]
- Daniel López Azaña, *Differences between physical CPU vs logical CPU vs Core vs Thread vs Socket* [Daniloaz.com]
- Manuel du programmeur Linux :
  - *uname (1)* [man]
  - *free (1)* [man]
  - *df (1)* [man]
  - *who* (1) [man]
- Christopher Murray, *Understanding Crontab in Linux With Examples* [Linux Handbook]
- Wikipédia, *cron* [Wikipédia]

APT crontab cursus 42 Debian machine virtuelle ordinateur SSH sudo UFW

A PROPOS DE L'AUTEUR

Mia Combeau

Étudiante à 42Paris, exploratrice du monde numérique. Je code ici à la norme de l'école 42, donc les boucles for, les switch, les opérateurs ternaires et plein d'autres choses me sont hors de portée... pour l'instant !

AUTRES ARTICLES

LAISSER UN COMMENTAIRE

Commentaire

Nom *

E-mail *

Site web

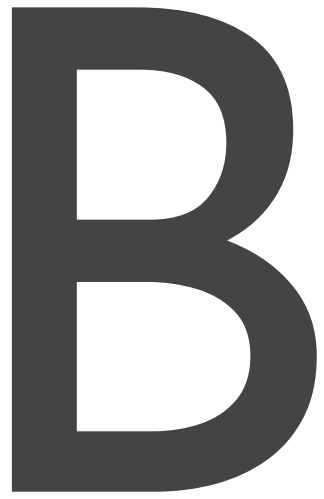
▢ Enregistrer mon nom, mon e-mail et mon site dans le navigateur pour mon prochain commentaire.

ENVOYER LE COMMENTAIRE

LIRE LA SUITE

Binaire 010 : l'utilité des opérations bitwise et du bit shifting

7 mai 2022



Pipex : reproduire l'opérateur pipe « | » en C

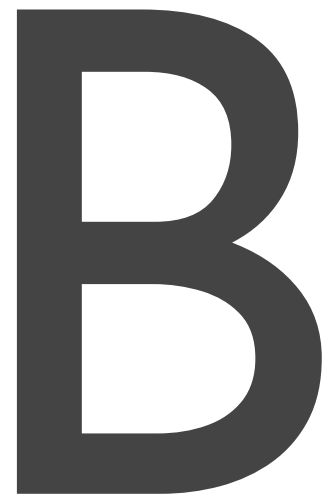
2 avril 2022

Born2beroot 03 : installer WordPress sur un serveur Debian

13 mars 2022

Born2beroot 01 : créer une machine virtuelle Debian

9 mars 2022



Par Mia Combeau 11 mars 2022
Remonter
Born2beroot 03 : installer WordPress sur un serveur Debian
Remonter
Born2beroot 01 : créer une machine virtuelle Debian

MENU

- Mentions légales
- Contact

CATÉGORIES

- Informatique
- Programmation en C
- Projets 42

codequoi

- ACCUEIL
- CATÉGORIES
  - PROGRAMMATION EN C
  - INFORMATIQUE
  - OUTILS DE PROGRAMMATION

CATÉGORIES

- Informatique
- Programmation en C
- Projets 42

MIA COMBEAU

Étudiante à 42Paris, exploratrice du monde numérique. Je code ici à la norme de l'école 42, donc les boucles for, les switch, les opérateurs ternaires et plein d'autres choses me sont hors de portée... pour l'instant !

- github
- linkedin