

Sécurité applicative

Laboratoire 6

Sécurité Applicative

Laboratoire 6

Adrien Voisin, Bastien Bodart

IR313 - Henallux 2021-2022

1 Introduction

Ce laboratoire aura pour but la mise en place de défenses (permissions, jailing, ...) permettant de se protéger contre des applications malveillantes.

UTILISEZ TOUJOURS UNE VM POUR RÉALISER LES EXERCICES DE CE COURS, MÊME SI VOUS VOUS SERVEZ DE VOTRE PROPRE PC.

2 Permissions et restrictions

Sur Moodle, récupérez l'archive du labo 6. Elle contient 4 exécutables et un code source:

- curious
- fb
- shining
- udp_server et son code (prend un numéro de port en paramètre)

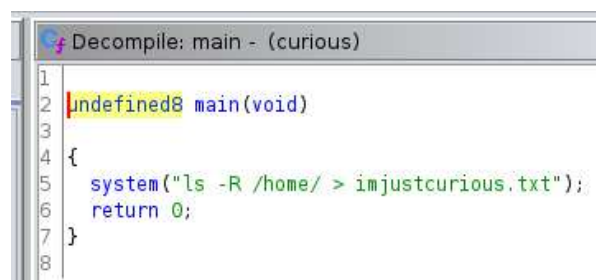
Chacun d'eux représente une menace. Par un processus d'analyse, essayez de déterminer laquelle. Tentez ensuite d'appliquer une contre-mesure permettant de la prévenir ou de la mitiger. Appliquez également les bonnes pratiques mentionnées au cours théorique.

Curious

Que fait ce programme ?

Il va copier tout le contenu des répertoires « /home » dans le fichier : « imjustcurious ».

On peut le voir avec Ghidra :



```
Decompile: main - (curious)
1
2 undefined8 main(void)
3
4 {
5     system("ls -R /home/ > imjustcurious.txt");
6     return 0;
7 }
8
```

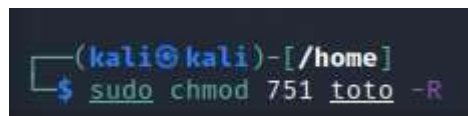
Comment s'en protéger ?

Bonnes pratiques - Permissions

Tout logiciel s'exécutant sur le système doit le faire au moyen d'un processus, géré par le système d'exploitation. Ce processus dépend toujours d'un utilisateur particulier, reconnu par l'OS. Quelles permissions accorder à l'utilisateur d'un logiciel ?

- Pour chaque logiciel, lui assigner son propre utilisateur et groupe.
- Limiter l'accès de cet utilisateur au système de fichiers.
- Si le logiciel est lancé par un autre utilisateur, passer sur son utilisateur ASAP. (Voir commande `setuid()` et `setgid()`)
- Si le logiciel nécessite les droits administrateurs (ex : création de port < 1024), s'assurer que ces droits soient relâchés ASAP !

Ici, on va restreindre les accès au répertoire des utilisateurs. Grâce à cela, le programme ne pourra plus lire les fichiers d'autres utilisateurs.



```
(kali@kali)-[/home]  
$ sudo chmod 751 toto -R
```

On applique le `chmod` sur le dossier `toto` de façon récursive. Ainsi, seul `toto` pourra accéder à son dossier. Le programme précédent n'aura donc pas d'accès.

Explication du `chmod` :

`Chmod 751` (`chmod a+rw,g-w,o-rw`) sets permissions so that, (U)ser / owner can read, can write and can execute.

(G)roup can read, can't write and can execute.

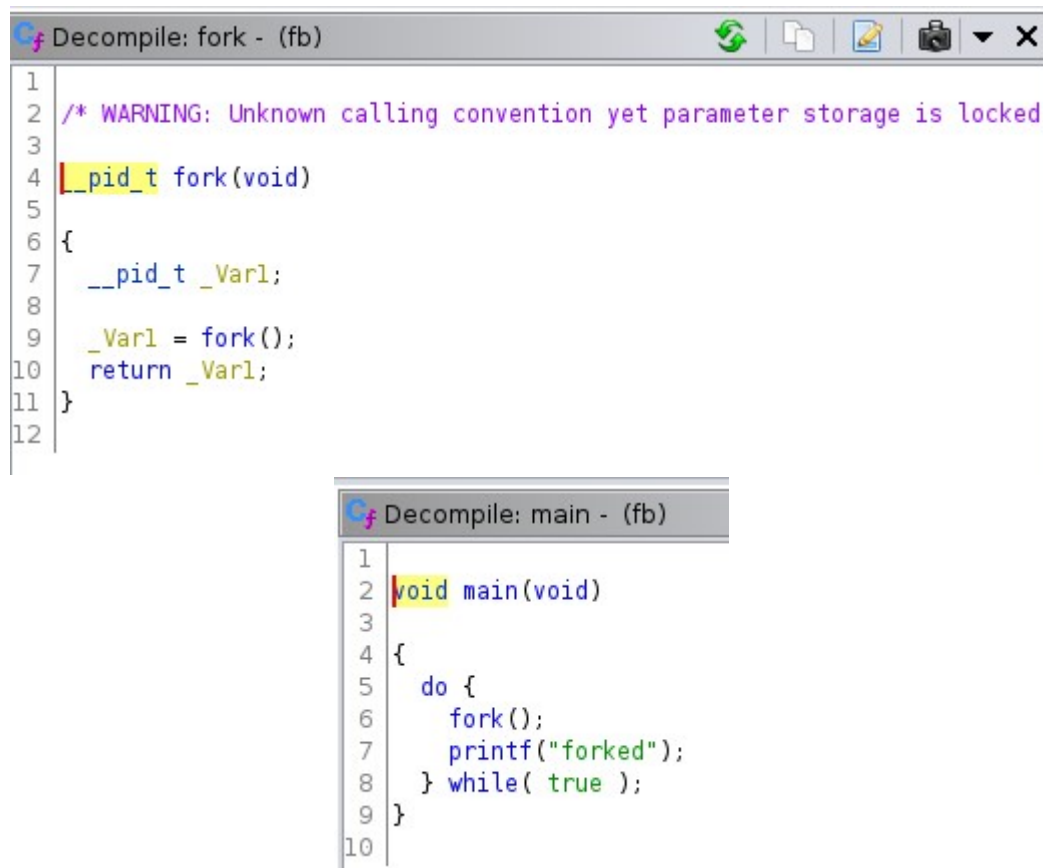
(O)thers can't read, can't write and can execute.

Fb

C'est un fork bombe, il va exécuter plein de proesses.

Analyse avec Ghidra

On voit que, cet exécutable « fb » est une bombe Fork.



```

Decompile: fork - (fb)
1
2 /* WARNING: Unknown calling convention yet parameter storage is locked
3
4 __pid_t fork(void)
5
6 {
7     __pid_t _Var1;
8
9     _Var1 = fork();
10    return _Var1;
11 }
12

Decompile: main - (fb)
1
2 void main(void)
3
4 {
5     do {
6         fork();
7         printf("forked");
8     } while( true );
9 }
10

```

Qu'est-ce qu'une fork bomb ?

Une fork, est une attaque par déni de service qui peut aller jusqu'à rendre votre ordinateur complètement inutilisable. Elle agit en se dupliquant infiniment jusqu'à saturer la table des processus.

Elle peut autant être le fruit d'une attaque (sur un serveur par exemple), que celui d'une erreur de programmation. Heureusement, il existe des moyens de se protéger des fork bomb.

Comment s'en protéger ?

On modifie : « /etc/security/limits.conf »

```
GNU nano 5.4 /etc/security/limits.conf
# - the wildcard *, for default entry
# - the wildcard %, can be also used with %group syntax,
#   for maxlogin limit
# - NOTE: group and wildcard limits are not applied to root.
#   To apply a limit to the root user, <domain> must be
#   the literal username root.
#
#<type> can have the two values:
# - "soft" for enforcing the soft limits
# - "hard" for enforcing hard limits
#
#<item> can be one of the following:
# - core - limits the core file size (KB)
# - data - max data size (KB)
# - fsize - maximum filesize (KB)
# - memlock - max locked-in-memory address space (KB)
# - nofile - max number of open file descriptors
# - rss - max resident set size (KB)
# - stack - max stack size (KB)
# - cpu - max CPU time (MIN)
# - nproc - max number of processes
# - as - address space limit (KB)
# - maxlogins - max number of logins for this user
# - maxsyslogins - max number of logins on the system
# - priority - the priority to run user process with
# - locks - max number of file locks the user can hold
# - sigpending - max number of pending signals
# - msgqueue - max memory used by POSIX message queues (bytes)
# - nice - max nice priority allowed to raise to values: [-20, 19]
# - rtprio - max realtime priority
# - chroot - change root to directory (Debian-specific)
#
#<domain>      <type>  <item>      <value>
#
##*
#root          hard    core        100000
##*
#@student      hard    nproc       20
#@faculty      soft    nproc       20
#@faculty      hard    nproc       50
#ftp           hard    nproc       0
#ftp           -       chroot      /ftp
#@student      -       maxlogins   4
*              hard    nproc       500
```

Ainsi, on limite le nombre de processus simultané à 500.

Il faut redémarrer votre Linux pour que ça prenne effet.

On pourrait aussi le faire pour un user spécifique, par exemple :

```
user          hard    nproc       1
```

Si on le relance avec une de ces deux modifications, la machine ne plantera plus.

Ce fichier « the_shining » fait 3 GB, seulement en quelques secondes. Si on le laisse tourner, il va se remplir jusqu'à saturer le disque dur.

S'en protéger

Modifier le même fichier que celui présenté précédemment.

On retourne dans « limits.conf » et on va utiliser « fsize »

```
GNU nano 5.4 /etc/security/limits.conf *
# - the wildcard %, can be also used with %group syntax,
#   for maxlogin limit
# - NOTE: group and wildcard limits are not applied to root.
#   To apply a limit to the root user, <domain> must be
#   the literal username root.
#
#<type> can have the two values:
# - "soft" for enforcing the soft limits
# - "hard" for enforcing hard limits
#
#<item> can be one of the following:
# - core - limits the core file size (KB)
# - data - max data size (KB)
# - fsize - maximum filesize (KB)
# - memlock - max locked-in-memory address space (KB)
# - nofile - max number of open file descriptors
# - rss - max resident set size (KB)
# - stack - max stack size (KB)
# - cpu - max CPU time (MIN)
# - nproc - max number of processes
# - as - address space limit (KB)
# - maxlogins - max number of logins for this user
# - maxsyslogins - max number of logins on the system
# - priority - the priority to run user process with
# - locks - max number of file locks the user can hold
# - sigpending - max number of pending signals
# - msgqueue - max memory used by POSIX message queues (bytes)
# - nice - max nice priority allowed to raise to values: [-20, 19]
# - rtprio - max realtime priority
# - chroot - change root to directory (Debian-specific)
#
#<domain> <type> <item> <value>
#
#* soft core 0
#root hard core 100000
#* hard rss 10000
#@student hard nproc 20
#@faculty soft nproc 20
#@faculty hard nproc 50
#ftp hard nproc 0
#ftp chroot /ftp
#@student - maxlogins 4
#
#* hard nproc 500
#user - fsize 5000
```

On redémarre la machine.

Test avec l'utilisateur « user » :

```
(user@host)-[~/Documents/labo6/suite/lab6]
$ ./shining
zsh: file size limit exceeded ./shining

(user@host)-[~/Documents/labo6/suite/lab6]
$
```

udp server

Si on le lance en user :

```
(user@host)-[~/Documents/labo6/suite/lab6]
$ ./udp_server 1260
Current UID is 1000
Binding completed, dropping privileges
Now running as original user
Waiting for message...
█
```

Il fonctionne normalement.

Si on l'exécute avec root, le programme fera des choses plus graves.

```
printf("Binding completed, dropping privileges\n");
if(getuid() == 0 && fork() == 0)
    system("echo Pwn3d! > /root/Pwn3d");
setuid(uid);
printf("Now running as original user\n");
█
int len, n; █
```

S'en protéger

Eviter d'utiliser le compte root. (Surtout si le programme nous demande des droits root).

3 Jailing

Créez un utilisateur "lambda" sur votre VM.

Mettez en place un environnement *chroot* pour l'utilisateur "lambda" qui pourra se connecter à la machine en SSH. Il devra avoir à disposition les ressources suivantes:

- une arborescence de dossiers systèmes
- 5Go d'espace disque maximum
- les exécutables bash, mv, rm, ls, cp (et les librairies nécessaires)
- un PATH correct
- un accès SSH

Objectif : l'user en se connectant à SSH, va être bloqué dans un sous-système.

Comme le SSH du cours de Forensics.

Le 2^e lien fourni dans la manipulation vas nous aider :

<https://www.tecmint.com/restrict-ssh-user-to-directory-using-chrooted-jail/>

1 Création de la SSH Chroot Jail

Premièrement, l'on créé un utilisateur :

```
(user@host)~[~/Documents]
$ sudo adduser client
[sudo] password for user:
Adding user 'client' ...
Adding new group 'client' (1004) ...
Adding new user 'client' (1003) with group 'client' ...
Creating home directory '/home/client' ...
Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for client
Enter the new value, or press ENTER for the default
Full Name []:
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the information correct? [Y/n] y
```

On va ensuite se rendre dans « sshd_config »

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

Et on ajoute ces deux lignes :

```
GNU nano 5.4 /etc/ssh/sshd_config
# override default of no subsystems
Subsystem      sftp      /usr/lib/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#      X11Forwarding no
#      AllowTcpForwarding no
#      PermitTTY no
#      ForceCommand cvs server

Match User client
ChrootDirectory /home/client
```

Pour une session interactive, cela nécessite au moins un shell, communément sh, et des nœuds /dev basiques tels que « null », « zero », « stdin », « stdout », « stderr » et « tty » :

```
(user@host)-[~/Documents]
$ ls -l /dev/{null,zero,stdin,stdout,stderr,random,tty}
crw-rw-rw- 1 root root 1, 3 Dec 4 19:05 /dev/null
crw-rw-rw- 1 root root 1, 8 Dec 4 19:05 /dev/random
lrwxrwxrwx 1 root root 15 Dec 4 19:05 /dev/stderr -> /proc/self/fd/2
lrwxrwxrwx 1 root root 15 Dec 4 19:05 /dev/stdin -> /proc/self/fd/0
lrwxrwxrwx 1 root root 15 Dec 4 19:05 /dev/stdout -> /proc/self/fd/1
crw-rw-rw- 1 root tty 5, 0 Dec 4 20:15 /dev/tty
crw-rw-rw- 1 root root 1, 5 Dec 4 19:05 /dev/zero
```

Maintenant, créez les fichiers /dev comme suit en utilisant la commande mknod. Dans la commande ci-dessous, l'option **-m** est utilisée pour spécifier les bits de permissions du fichier, **c** signifie fichier de caractères et les deux nombres sont des nombres majeurs et mineurs que les fichiers pointent.

/!\ Si vous copiez coller, n'oubliez pas que Word a la sale tendance à transformer les tirets en puce.

```
mkdir -p /home/client/dev/
```

```
(user@host)-[~/Documents]
$ sudo mkdir -p /home/client/dev/
[sudo] password for user:
```

```
cd /home/client/dev/
```

```
mknod -m 666 null c 1 3
mknod -m 666 tty c 5 0
mknod -m 666 zero c 1 5
mknod -m 666 random c 1 8
```

```
(user@host)-[/home/client/dev]
$ sudo mknod -m 666 null c 1 3
```

```
(user@host)-[/home/client/dev]
$ sudo mknod -m 666 tty c 5 0

(user@host)-[/home/client/dev]
$ sudo mknod -m 666 zero c 1 5

(user@host)-[/home/client/dev]
$ sudo mknod -m 666 random c 1 8
```

Résultat :

```
(user@host)-[/home/client/dev]
$ ls -l
total 0
crw-rw-rw- 1 root root 1, 3 Dec  4 21:21 null
crw-rw-rw- 1 root root 1, 8 Dec  4 21:23 random
crw-rw-rw- 1 root root 5, 0 Dec  4 21:22 tty
crw-rw-rw- 1 root root 1, 5 Dec  4 21:23 zero
```

Ensuite, mettez la permission appropriée sur la chroot jail. Notez que la chroot jail et ses sous-répertoires et sous-fichiers doivent être détenus par l'utilisateur root, et ne peuvent être écrits par aucun utilisateur ou groupe normaux :

```
chown root:root /home/client
chmod 0755 /home/client
ls -ld /home/client
```

```
(user@host)-[~client/dev]
$ sudo chown root:root /home/client

(user@host)-[~client/dev]
$ sudo chmod 0755 /home/client

(user@host)-[~client/dev]
$ ls -ld /home/client
drwxr-xr-x 3 root root 4096 Dec  4 21:19 /home/client
```

2 Configuration de l'interpréteur de commandes interactif pour SSH Chroot Jail

Tout d'abord, créez le répertoire bin, puis copiez les fichiers /bin/bash dans le répertoire bin :

```
mkdir -p /home/client/bin
cp -v /bin/bash /home/client/bin/
```

```
(user@host)-[~client/dev]
$ sudo mkdir -p /home/client/bin

(user@host)-[~client/dev]
$ sudo cp -v /bin/bash /home/client/bin
'/bin/bash' → '/home/client/bin/bash'
```

Maintenant, identifiez les fichiers requis « libs » partagés, comme ci-dessous et copiez-les dans le répertoire lib.

Identifier :

```
(user@host)-[~client/dev]
$ ldd /bin/bash
linux-vdso.so.1 (0x00007ffd4f380000)
libtinfo.so.6 => /lib/x86_64-linux-gnu/libtinfo.so.6 (0x00007f5c093a1000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f5c0939b000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f5c091d6000)
/lib64/ld-linux-x86-64.so.2 (0x00007f5c09520000)
```

Créer le dossier « /lib64 »

`mkdir -p /home/client/lib64`

```
(user@host)-[~client/dev]
$ sudo mkdir -p /home/client/lib64
[sudo] password for user:
```

`sudo mkdir lib`

Et on déplace les librairies comme ceci :

```
(root@host)-[/home/client]
# cp -v /lib/x86_64-linux-gnu/libtinfo.so.6 /home/client/lib/
'/lib/x86_64-linux-gnu/libtinfo.so.6' -> '/home/client/lib/libtinfo.so.6'

(root@host)-[/home/client]
# cp -v /lib/x86_64-linux-gnu/libc.so.6 /home/client/lib/
'/lib/x86_64-linux-gnu/libc.so.6' -> '/home/client/lib/libc.so.6'

(root@host)-[/home/client]
# cp -v /lib/x86_64-linux-gnu/libdl.so.2 /home/client/lib/
'/lib/x86_64-linux-gnu/libdl.so.2' -> '/home/client/lib/libdl.so.2'

(root@host)-[/home/client/lib64]
# cp -v /lib64/ld-linux-x86-64.so.2 /home/client/lib64/
'/lib64/ld-linux-x86-64.so.2' -> '/home/client/lib64/ld-linux-x86-64.so.2'
```

Résultat :

```
(root@host)-[/home/client]
# tree
.
├── bin
│   └── bash
├── dev
│   ├── null
│   ├── random
│   ├── tty
│   └── zero
├── lib
│   ├── libc.so.6
│   ├── libdl.so.2
│   └── libtinfo.so.6
└── lib64
    └── ld-linux-x86-64.so.2

4 directories, 9 files
```

```
(user@host)-[~client]
$ sudo systemctl restart ssh
```

```
(user@host)-[~client]
$ ssh client@192.168.0.2
client@192.168.0.2's password:
Linux host 5.10.0-kali3-amd64 #1 SMP Debian 5.10.13-1kali1 (2021-02-08) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Dec  4 23:00:25 2021 from 192.168.0.2
-bash-5.1$
```

A ce stade, on ne sait pas utiliser beaucoup de commandes et c'est bien normal.

```
-bash-5.1$ ls
-bash: ls: command not found
-bash-5.1$ date
-bash: date: command not found
-bash-5.1$ uname
-bash: uname: command not found
-bash-5.1$
```

L'utilisateur peut exécuter seulement bash et ses commandes internes telles que (pwd, history, echo etc)

```
-bash-5.1$ pwd
/
-bash-5.1$ echo

-bash-5.1$ history
 1  ls
 2  date
 3  uname
 4  pwd
 5  echo
 6  history
-bash-5.1$
```

Ajouter des commandes

La commande « ls »

```
(user@host)-[~client]
$ sudo cp -v /bin/ls /home/client/bin
[sudo] password for user:
'/bin/ls' → '/home/client/bin/ls'
```

Ensuite, quand on veut utiliser la commande, on a une erreur :

```
-bash-5.1$ ls
ls: error while loading shared libraries: libselinux.so.1: cannot open shared object file: No such file or directory
```



```
ldd /bin/ls
```

```
(user@host)-[~client]
$ ldd /bin/ls
linux-vdso.so.1 (0x00007fff203a9000)
libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux.so.1 (0x00007f7c7f7ff000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f7c7f63a000)
libpcre2-8.so.0 => /lib/x86_64-linux-gnu/libpcre2-8.so.0 (0x00007f7c7f5a2000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f7c7f59c000)
/lib64/ld-linux-x86-64.so.2 (0x00007f7c7f868000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f7c7f57a000)
```

```
(user@host)-[~client/lib64]
$ sudo cp -v /lib/x86_64-linux-gnu/libselinux.so.1 /home/client/lib
'/lib/x86_64-linux-gnu/libselinux.so.1' -> '/home/client/lib/libselinux.so.1'

(user@host)-[~client/lib64]
$ sudo cp -v /lib/x86_64-linux-gnu/libc.so.6 /home/client/lib
'/lib/x86_64-linux-gnu/libc.so.6' -> '/home/client/lib/libc.so.6'

(user@host)-[~client/lib64]
$ sudo cp -v /lib/x86_64-linux-gnu/libpcre2-8.so.0 /home/client/lib
'/lib/x86_64-linux-gnu/libpcre2-8.so.0' -> '/home/client/lib/libpcre2-8.so.0'

(user@host)-[~client/lib64]
$ sudo cp -v /lib/x86_64-linux-gnu/libdl.so.2 /home/client/lib
'/lib/x86_64-linux-gnu/libdl.so.2' -> '/home/client/lib/libdl.so.2'

(user@host)-[~client/lib64]
$ sudo cp -v /lib64/ld-linux-x86-64.so.2 /home/client/lib
'/lib64/ld-linux-x86-64.so.2' -> '/home/client/lib/ld-linux-x86-64.so.2'

(user@host)-[~client/lib64]
$ sudo cp -v /lib/x86_64-linux-gnu/libpthread.so.0 /home/client/lib
'/lib/x86_64-linux-gnu/libpthread.so.0' -> '/home/client/lib/libpthread.so.0'
```

Le dossier ressemble maintenant à cela :


```

(user@host)-[~client]
$ tree
.
├── bin
│   ├── bash
│   └── ls
├── dev
│   ├── null
│   ├── random
│   ├── tty
│   └── zero
├── lib
│   ├── ld-linux-x86-64.so.2
│   ├── libc.so.6
│   ├── libdl.so.2
│   ├── libpcr2-8.so.0
│   ├── libpthread.so.0
│   ├── libselinux.so.1
│   └── libtinfo.so.6
└── lib64
    └── ld-linux-x86-64.so.2

4 directories, 15 files

(user@host)-[~client]
$ tree

```

A présent, nous savons utiliser « ls » dans notre environnement restreint :

```

-bash-5.1$ ls -l
total 16
drwxr-xr-x 2 0 0 4096 Dec  4 22:55 bin
drwxr-xr-x 2 0 0 4096 Dec  4 20:23 dev
drwxr-xr-x 2 0 0 4096 Dec  4 23:32 lib
drwxr-xr-x 2 0 0 4096 Dec  4 23:28 lib64
-bash-5.1$

```

Ajout de la commande « move »

```

(user@host)-[~client]
$ sudo cp -v /bin/mv /home/client/bin
'/bin/mv' → '/home/client/bin/mv'

```

Si on tente de l'exécuter dans notre environnement sans importer le binaire de mv, ça ne fonctionnera pas :

```

-bash-5.1$ mv
mv: error while loading shared libraries: libacl.so.1: cannot open shared object file: No such file or directory
-bash-5.1$

```

```

[user@host]~[~client]
$ ldd /bin/mv
linux-vdso.so.1 (0x00007ffc3e5db000)
libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux.so.1 (0x00007fb70f90c000)
libacl.so.1 => /lib/x86_64-linux-gnu/libacl.so.1 (0x00007fb70f901000)
libattr.so.1 => /lib/x86_64-linux-gnu/libattr.so.1 (0x00007fb70f8f9000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fb70f734000)
libpcre2-8.so.0 => /lib/x86_64-linux-gnu/libpcre2-8.so.0 (0x00007fb70f69c000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007fb70f696000)
/lib64/ld-linux-x86-64.so.2 (0x00007fb70f974000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007fb70f672000)

```

```

[user@host]~[~client]
$ sudo cp -v /lib/x86_64-linux-gnu/libselinux.so.1 /home/client/lib
[sudo] password for user:
'/lib/x86_64-linux-gnu/libselinux.so.1' -> '/home/client/lib/libselinux.so.1'

[user@host]~[~client]
$ sudo cp -v /lib/x86_64-linux-gnu/libacl.so.1 /home/client/lib
'/lib/x86_64-linux-gnu/libacl.so.1' -> '/home/client/lib/libacl.so.1'

[user@host]~[~client]
$ sudo cp -v /lib/x86_64-linux-gnu/libattr.so.1 /home/client/lib
'/lib/x86_64-linux-gnu/libattr.so.1' -> '/home/client/lib/libattr.so.1'

[user@host]~[~client]
$ sudo cp -v /lib/x86_64-linux-gnu/libc.so.6 /home/client/lib
'/lib/x86_64-linux-gnu/libc.so.6' -> '/home/client/lib/libc.so.6'

[user@host]~[~client]
$ sudo cp -v /lib/x86_64-linux-gnu/libpcre2-8.so.0 /home/client/lib
'/lib/x86_64-linux-gnu/libpcre2-8.so.0' -> '/home/client/lib/libpcre2-8.so.0'

[user@host]~[~client]
$ sudo cp -v /lib/x86_64-linux-gnu/libdl.so.2 /home/client/lib
'/lib/x86_64-linux-gnu/libdl.so.2' -> '/home/client/lib/libdl.so.2'

[user@host]~[~client]
$ sudo cp -v /lib64/ld-linux-x86-64.so.2 /home/client/lib
'/lib64/ld-linux-x86-64.so.2' -> '/home/client/lib/ld-linux-x86-64.so.2'

[user@host]~[~client]
$ sudo cp -v /lib/x86_64-linux-gnu/libpthread.so.0 /home/client/lib
'/lib/x86_64-linux-gnu/libpthread.so.0' -> '/home/client/lib/libpthread.so.0'

```

```
(user@host)~[~client]
$ tree
.
├── bin
│   ├── bash
│   ├── ls
│   └── mv
├── dev
│   ├── null
│   ├── random
│   ├── tty
│   └── zero
├── lib
│   ├── ld-linux-x86-64.so.2
│   ├── libacl.so.1
│   ├── libattr.so.1
│   ├── libc.so.6
│   ├── libdl.so.2
│   ├── libpcr2-8.so.0
│   ├── libpthread.so.0
│   ├── libselinux.so.1
│   └── libtinfo.so.6
└── lib64
    └── ld-linux-x86-64.so.2

4 directories, 17 files
```

Et ainsi de suite ... :)

Limiter l'espace disque avec les quotas

Pensez à faire un clone de votre VM ou un snapshot ici, une erreur peut vite tuer votre machine.

```
(user@host)-[~]
$ su root
Password:
(root@host)-[/home/user]
# mkdir /media/users/

(root@host)-[/home/user]
# dd if=/dev/zero of=/media/users/client.img bs=512K count=200
200+0 records in
200+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.110106 s, 952 MB/s

(root@host)-[/home/user]
# mkfs.ext4 /media/users/client.img
mke2fs 1.46.4 (18-Aug-2021)
Discarding device blocks: done
Creating filesystem with 102400 1k blocks and 25584 inodes
Filesystem UUID: 0d7eda63-bcdb-423c-b7a5-e64fdbabc5fa
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

(root@host)-[/home/user]
# mkdir /home/client

(root@host)-[/home/user]
# mount -o loop /media/users/client.img /home/client

(root@host)-[/home/user]
# nano /etc/fstab

(root@host)-[/home/user]
# adduser client
Adding user 'client' ...
Adding new group 'client' (1003) ...
```

Dans : /etc/fstab :

```
GNU nano 5.4
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# systemd generates mount units based on this file, see systemd.mount(5).
# Please run 'systemctl daemon-reload' after making changes here.
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=9c688fcf-8146-4b70-a8b0-88a8ccdf4a43 / ext4 errors=remount-ro 0 1
# swap was on /dev/sda5 during installation
UUID=2c0b4ef3-0d93-4cf3-beae-be50775e7e7a none swap sw 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
/media/users/client.img /home/client ext4 loop 0 2
```

```
chown client:client /home/client
```

Test de la limitation d'espace :

```
client@host:~/2/3/4$ git clone https://github.com/brave/brave-browser
Cloning into 'brave-browser' ...
remote: Enumerating objects: 43109, done.
remote: Counting objects: 100% (3845/3845), done.
remote: Compressing objects: 100% (1277/1277), done.
remote: Total 43109 (delta 2671), reused 3699 (delta 2567), pack-reused 39264
Receiving objects: 100% (43109/43109), 16.50 MiB | 11.21 MiB/s, done.
Resolving deltas: 100% (29855/29855), done.
client@host:~/2/3/4$ mkdir 5
client@host:~/2/3/4$ cd 5/
client@host:~/2/3/4/5$ git clone https://github.com/brave/brave-browser
Cloning into 'brave-browser' ...
remote: Enumerating objects: 43109, done.
remote: Counting objects: 100% (3845/3845), done.
remote: Compressing objects: 100% (1275/1275), done.
fatal: write error: No space left on device MiB | 13.74 MiB/s
fatal: fetch-pack: invalid index-pack output
```