# Couteau suisse du confinement sous Linux

Vincent Autefage

vincent.autefage@u-bordeaux.fr

Informatique / iut de BORDEAUX


université de BORDEAUX

# Principe du moindre privilège

## Définition de l'ANSSI

Un mécanisme de **cloisonnement** permet de **compartimenter** un **environnement d'exécution** en plusieurs parties ne comportant pas les **mêmes éléments** et ne bénéficiant ni des **mêmes droits** ni des **mêmes ressources**.

Une fois ceci mis en œuvre, la **compromission** d'une sous-partie devient **plus difficile** car sa **surface d'attaque** est **réduite**.

# Isolation de processus

## Objectif

**Restreindre l'accès au système** et aux **ressources** pour un **processus unique** ou bien un **groupe de processus**.

Exemples :

- Restriction d'accès au **système de fichiers**
- Limitation sur l'**utilisation des ressources** comme le CPU ou la RAM
- Limitation sur l'accès à certains **périphériques matériels**

# État de l'art - chroot

Apparu à la fin des années 1970, chroot est un **appel système** permettant de **changer la racine** du système de fichiers.

```
root@valtax[vfs]$ pwd
/tmp/vfs
root@valtax[vfs]$ ls
debian11root
root@valtax[vfs]$ ls debian11root/
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@valtax[vfs]$ chroot debian11root/
root@valtax:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@valtax:/# pwd
/
root@valtax:/#
```

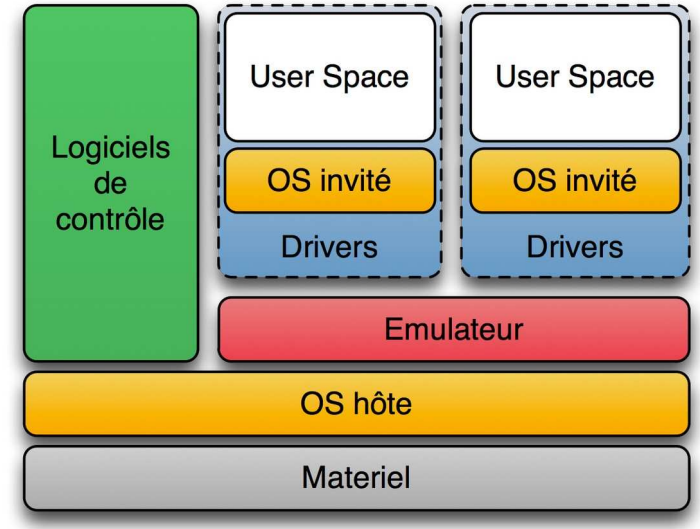
# État de l'art – Jails BSD

Apparus au début des années 2000, les Jails BSD introduisent le concept de **conteneur** proposant une **isolation multi-niveaux** :

- Isolation du **système de fichiers**
- Isolation des **utilisateurs**
- Isolation des **processus**
- Isolation **réseau**
- Partage du **même noyau**

# État de l'art – Émulation / Virtualisation

Les machines virtuelles émulent tout ou partie du matériel dans le but de faire tourner en **espace utilisateur** une **image disque complète** incluant son **propre noyau**.




Source : Wikimedia

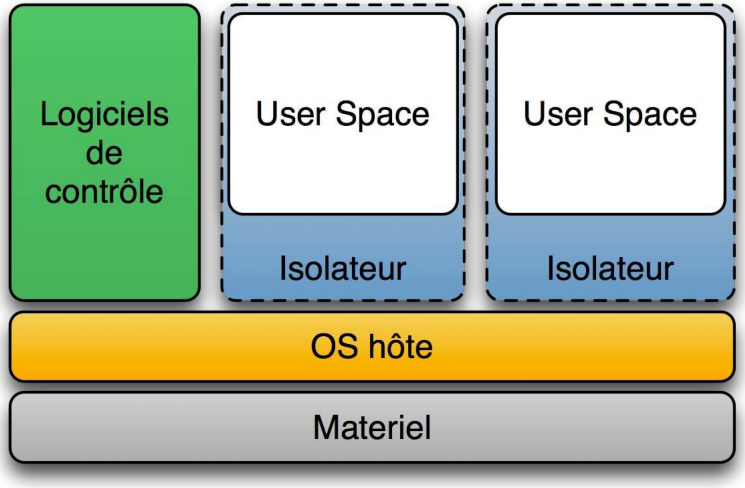
# État de l'art – Conteneurs / Isolateurs

Les conteneurs sont l'aboutissement actuel des **Jails BSD** centrés sur des **images disques** contenant un **système de fichiers dédié.**

cri-o

[ ● ◂ ] systemd

LXC




Source : Wikimedia

docker

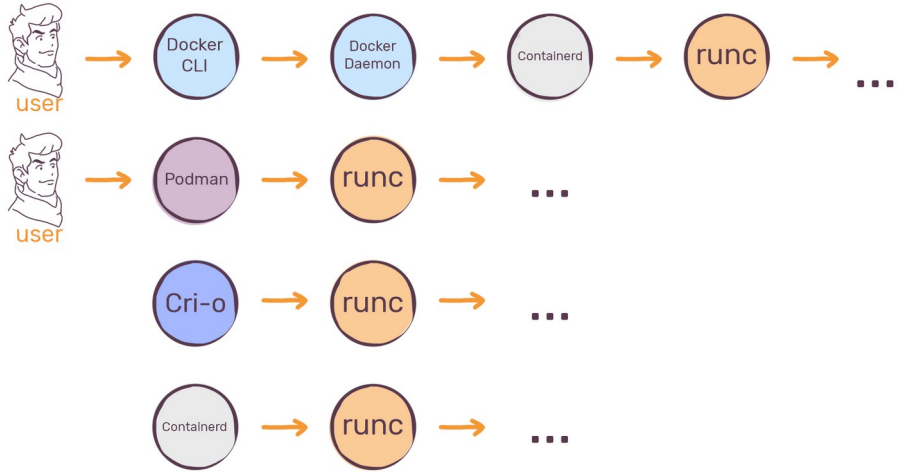
podman

containerd

# État de l'art – Conteneurs - Engine vs Runtime

Le moteur d'exécution (runtime) est chargé d'**instancier** et de **gérer le cycle de vie** des conteneurs.

Le moteur de conteneur (engine) est l'**interface** qui permet à l'**utilisateur** de **manipuler le** runtime.

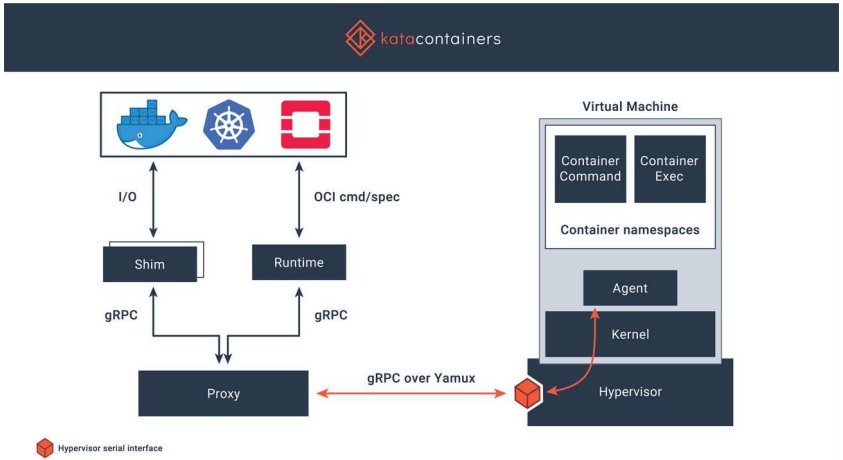




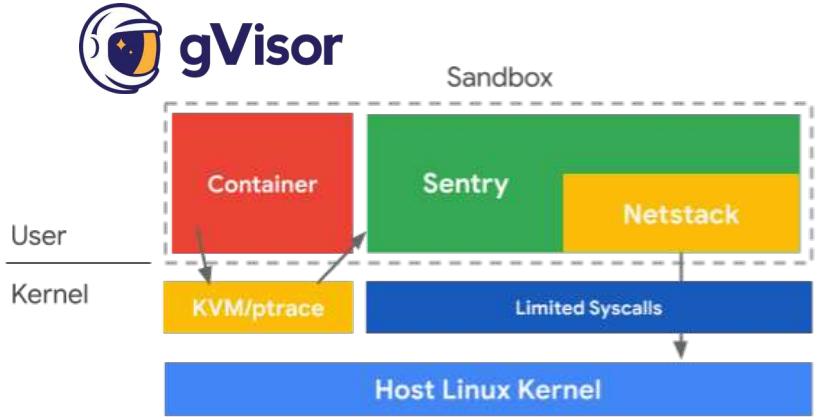
Source : mkdev.me

# État de l'art – Virtualisation légère

La virtualisation légère est à mi-chemin entre un conteneur et une machine virtuelle exploitant un **pseudo-noyau virtualisé** notamment pour la gestion des appels systèmes.



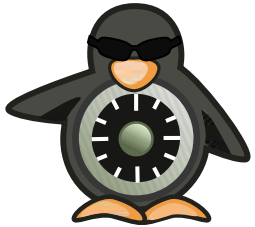

Source : katacontainers

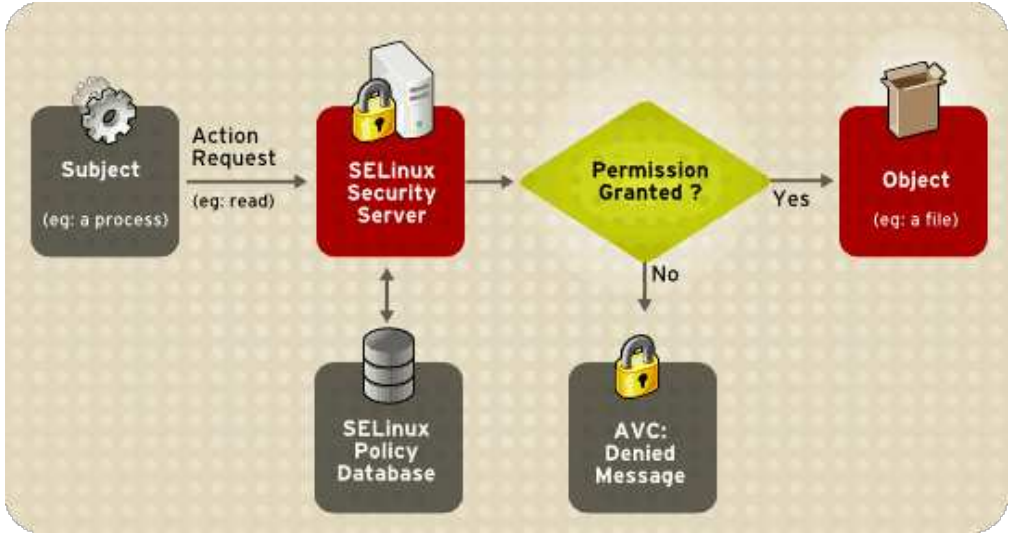



Source : gVisor

# État de l'art – Contrôle d'accès obligatoire

Linux Security Modules est un **framework** fourni par le **noyau Linux** permettant d'autoriser ou non un **processus** à accéder à une partie du système au travers d'un mécanisme de **hooks**.






Source : Red Hat

# Isolation des processus sous Linux

## Briques élémentaires de sécurité

- Namespaces
- Control Groups
- Capabilities
- Secure Computing Modes

# Namespaces

Linux 2.4

2002

Les namespaces permettent de créer un **groupe de processus** partageant des espaces de **ressources communes**.

Les groupes sont à la fois **isolés du système** mais aussi **les uns des autres**.

La manipulation des namespaces est permise au travers des appels systèmes clone, setns et unshare.

Il existe à l'heure actuelle 8 namespaces :

- pid
- ipc
- network
- user
- uts
- time
- mount
- cgroup

# Coté système

```
> ip -br addr
lo       UNKNOWN    127.0.0.1/8
eno1     UP         192.168.1.14/24

> ps -e
PID TTY          TIME CMD
...
11456 pts/1    00:00:00 bash
13611 pts/1    00:00:00 ps

> hostname
oh-my-linux

> unshare --uts --pid --net --fork \
--mount-proc /bin/bash
```

# Utilisation des namespaces


Lancer un nouveau **shell isolé sans accès réseaux**

# Dans l'antre du namespace

```
ns > ip -br addr
lo        DOWN

ns > ip addr add 127.1.2.3/8 dev lo
ns > ip -br addr
lo        UNKNOWN     127.1.2.3/8

ns > ps -e
PID TTY          TIME CMD
  1 pts/1    00:00:00 bash
 89 pts/1    00:00:00 ps

ns > hostname linux-in-ns
ns > hostname
linux-in-ns
```

## Utilisation des namespaces


Lancer un nouveau **shell isolé sans accès réseaux**

# La lumière au bout du tunnel

```
ns > exit

> ip -br addr
lo       UNKNOWN    127.0.0.1/8
eno1     UP         192.168.1.14/24

> ps -e
PID TTY          TIME CMD
...
11456  pts/1    00:00:00 bash
13627  pts/1    00:00:00 ps

> hostname
oh-my-linux
```

## Utilisation des namespaces


Lancer un nouveau **shell isolé sans accès réseaux**

# Control Groups

Linux 2.6/4.5

2007/2016

Les cgroups permettent de **limiter la consommation des ressources** du système pour un groupe de processus donné.

Les cgroups se configurent via des **controllers** définis dans les répertoires /sys/fs/cgroup et /proc/<pid>/cgroup.

Il existe **9** controllers **v2** principaux (**13** en **v1**) :

- cpu
- cpuset
- freezer
- hugelb
- io
- memory
- perf_event
- pids
- rdma

# Control Groups v2 - pids

```
> mount |grep cgroup
cgroup2 on /sys/fs/cgroup type cgroup2

> echo +pid > /sys/fs/cgroup/cgroup.subtree_control
> mkdir /sys/fs/cgroup/imb
> echo 2 > /sys/fs/cgroup/imb/pids.max
> echo $$ > /sys/fs/cgroup/imb/cgroup.procs
> whoami
root

> echo 1 > /sys/fs/cgroup/imb/pids.max
> whoami
sh: Cannot fork

> echo max > /sys/fs/cgroup/imb/pids.max
> rmdir > /sys/fs/cgroup/imb
```

## Utilisation des cgroups


Limiter le **nombre de processus fils**

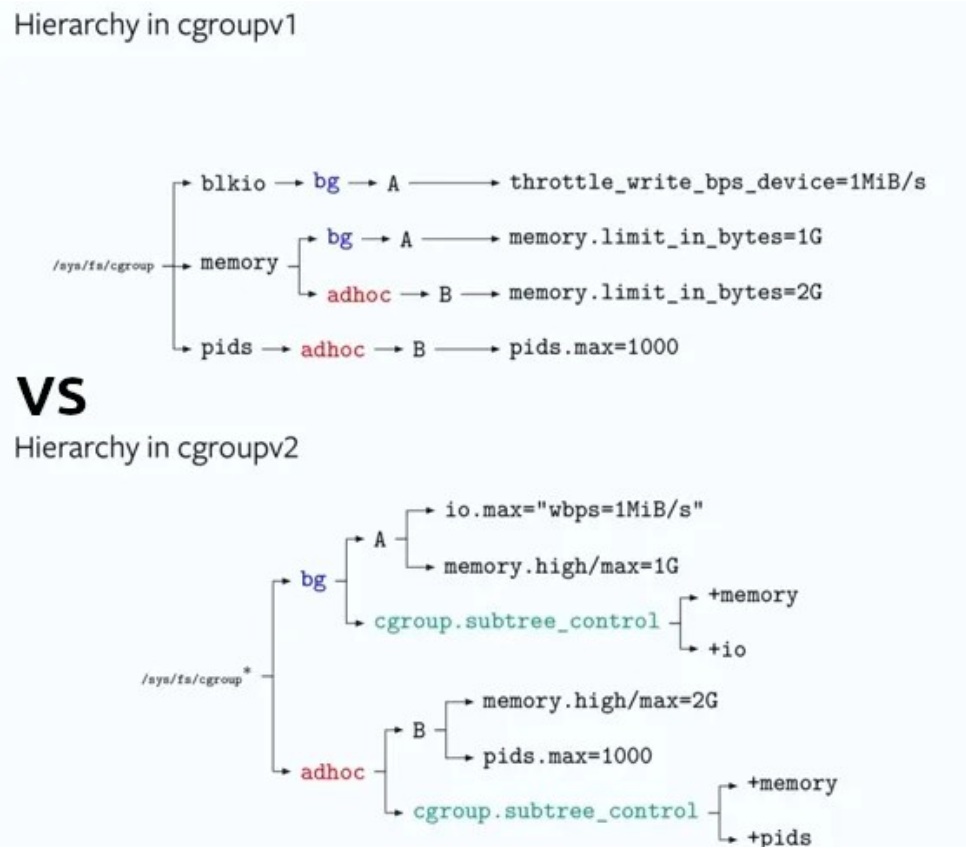
# Control Groups v1 - devices

```
> mount |grep cgroup
cgroup_root on /sys/fs/cgroup type tmpfs
cgroup on /sys/fs/cgroup/devices type cgroup
...

> dd if=/dev/random of=/tmp/blob bs=1M count=10
10485760 octets (10 MB, 10 MiB) copiés, 0,0071 s, 1,5 GB/s

> ls -l /dev/random
crw-rw-rw- 1 root root 1, 8 nov.  24 20:46 /dev/random

> mkdir /sys/fs/cgroup/devices/imb
> echo c 1:8 r > /sys/fs/cgroup/devices/imb/devices.deny
> echo $$ > /sys/fs/cgroup/devices/imb/cgroup.procs
> dd if=/dev/random of=/tmp/blob bs=1M count=10
dd: impossible d'ouvrir '/dev/random': Opération non permise

> echo > /sys/fs/cgroup/devices/imb/devices.deny
> rmdir /sys/fs/cgroup/devices/imb
```

# Utilisation des cgroups


Limiter l'**utilisation d'un périphérique**

# Control Groups – v1 versus v2




Source : Pavel Trukhanov

# Capabilities

Linux 2.2

1999

Les capabilities permettent de **limiter** ou **d'augmenter** les **privilèges** d'un processus.

La manipulation des capabilities est permise au travers des commandes setcap, getcap et setpriv.

Il existes une quarantaine de capabilities dont par exemple :

- cap_net_raw
- cap_net_bind_service
- cap_chown
- cap_mknod
- cap_sys_chroot
- cap_sys_module

# Une histoire de ping

```
> busybox ping -q -c1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
ping: permission denied (are you root?)

> getcap /bin/ping
/bin/ping cap_net_raw=ep

> sudo setcap "cap_net_raw+ep" /bin/busybox

> busybox ping -q -c1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
--- 127.0.0.1 ping statistics ---
1 transmitted, 1 received, 0% loss
round-trip min/avg/max = 0.070/0.070/0.070 ms
```

# Utilisation des capabilities


Autoriser un programme à ouvrir des **sockets en mode brute**

# Une histoire de port

```
> busybox httpd -f -vv -p 80
httpd: bind: Permission denied
```

```
> sudo setcap "cap_net_bind_service+ep" \
/bin/busybox
```

```
> busybox httpd -f -vv -p 80
Listening on :80
```

## Utilisation des capabilities


Autoriser un programme à utiliser des **ports privilégiés**

# Secure Computing Modes

Linux 2.6

2005

Les seccomps permettent à un processus de **supprimer** sa capacité à appeler certains **appels systèmes**.

La manipulation des seccomps est permise au travers des appels systèmes seccomp, bpf et prctl.

Le mode **filter** permet de sélectionner les appels systèmes concernés de façon exhaustive.

Un mode extrême dénommé **strict** permet de limiter la liste des appels systèmes pouvant être appelés à la liste suivante :

- read
- write
- _exit
- sigreturn

# Fichier Source en C

```c
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <linux/seccomp.h>
#include <sys/prctl.h>

int main(int argc, char **argv){
  int fd = open("out.txt", O_WRONLY | O_CREAT | O_SYNC);

  char * txt = "Hello There\n";
  write(fd, txt, strlen(txt)+1);

  prctl(PR_SET_SECCOMP, SECCOMP_MODE_STRICT);

  txt = "I'm still alive!\n";
  write(fd, txt, strlen(txt)+1);

  close(fd);

  fd = open("out.txt", O_RDONLY);
  printf("You will never see me!\n");
  close(fd);
  return EXIT_SUCCESS;
}
```

# Utilisation des seccomps


Illustration du **mode strict**

# Compilation & Exécution

```
> gcc -o strict strict.c

> cat out.txt
cat: out.txt: No such file or directory

> strace ./strict
execve("./strict", ["./strict"], 0x7ffd1f3789f0 /* 52 vars */) = 0
...
openat(AT_FDCWD, "out.txt", O_WRONLY|O_CREAT|O_SYNC, 023610) = 3
write(3, "Hello There\n\0", 13)         = 13
prctl(PR_SET_SECCOMP, SECCOMP_MODE_STRICT) = 0
write(3, "I'm still alive!\n\0", 18)    = 18
close(3)                                = ?
+++ killed by SIGKILL +++

> cat out.txt
Hello There
I'm still alive!
```

# Utilisation des seccomps


Illustration du **mode strict**

# Firejail – Un outil pour les gouverner tous

# Firejail – Un outil pour les gouverner tous

- Exécutable en ligne de commandes avec **setuid-bit**
- Écrit en **C sans dépendance**

  Disponible à partir de **Linux 3.0**
- **GPL v2**
- Création et gestion **simplifiées** de **sandbox**
- Repose sur les **namespaces**, les **cgroups**, les **capabilities** et les **seccomps**
- **Aucune image** ou système de fichiers dédié requis
- **Pas** de **démon** ou de **service** système

# Profil d'isolation

Un profil est une description de l'ensemble des restrictions formant la **sandbox**.

Un profil peut être défini **statiquement** sous la forme d'un **fichier** ou bien **dynamiquement** grâce à un ensemble d'options proposées par la **ligne de commande** de Firejail.

Firejail est fourni avec une **centaine de** profils localisés dans /etc/firejail.

Firejail se base sur le **nom de l'exécutable** afin d'appliquer un profil **particulier**.

Exemple : firejail firefox utilisera le profil défini dans /etc/firejail/firefox.profile.

# Shell privé non connecté

> firejail --noprofile --private –net=none bash

Lance une instance de bash avec les restrictions suivantes :

- arborescence des PIDs indépendante
- homedir mappé sur un répertoire temporaire vide
- cartes réseaux invisibles excepté la loopback

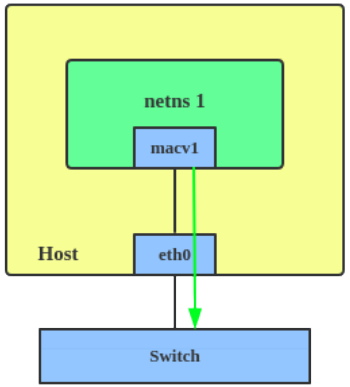
## Utilisation de Firejail


Lancer un nouveau **shell sans accès** au **homedir** et **sans accès réseaux**

# Capture réseau ciblée

```
> ip link add jail link eth0 type macvlan \
mode bridge

> ip addr add 192.168.42.1/24 dev jail \
> ip link set jail up

> echo 1 > /proc/sys/net/ipv4/ip_forward
> iptables -t nat -A POSTROUTING -s \
192.168.42.0/24 -o eth0 -j MASQUERADE

> wireshark -i jail -k

> firejail --noprofile --net=jail –dns=9.9.9.9 \
--defaultgw=192.168.42.1 bash
```

# Utilisation de Firejail

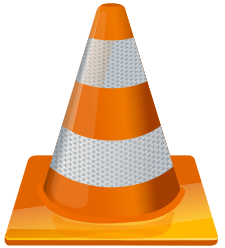



**Capturer** le **trafic réseau** des applications de la **sandbox**

# Profil par défaut de VLC (partie 1)

```
blacklist ${HOME}/.ssh
blacklist /etc/passwd
blacklist ${PATH}/python*

...

whitelist ${HOME}/.config/vlc
whitelist /var/tmp

...

noexec /tmp
noexec ${HOME}

...
```

# Utilisation de Firejail



**Profil verrouillant** les potentielles **actions illégitimes** de VLC

# Profil par défaut de VLC (partie 2)

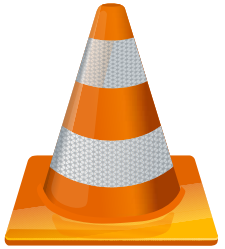
caps.drop all
seccomp
netfilter
protocol unix,inet,inet6,netlink

nogroups
nonewprivs
noroot
nou2f

private-bin cvlc,nvlc,qvlc,rvlc,svlc,vlc
private-dev
private-tmp

shell none
dbus-user none
dbus-system none
memory-deny-write-execute

## Utilisation de Firejail



**Profil verrouillant** les potentielles **actions illégitimes** de VLC

# Conclusion

- **Firejail** est une solution simple de création de **sandbox sur mesure**
- Il fournit un **accès simplifié** aux fonctionnalités de sécurité du noyau Linux
- Il ne nécessite **aucune image disque dédiée**
- Il a un **impact nul** sur les performances

## Exemples d'utilisation réelle

- Isolation de serveurs VNC dans un contexte industriel
- Plateforme d'expérimentation réseau
- Durcissement d'un système d'intégration continue

www.u-bordeaux.fr

@univbordeaux

@universitedebordeaux

@univbordeaux

@universitedebordeaux

Appli mobile U&me

@univbordeaux

Merci de votre attention

université de BORDEAUX