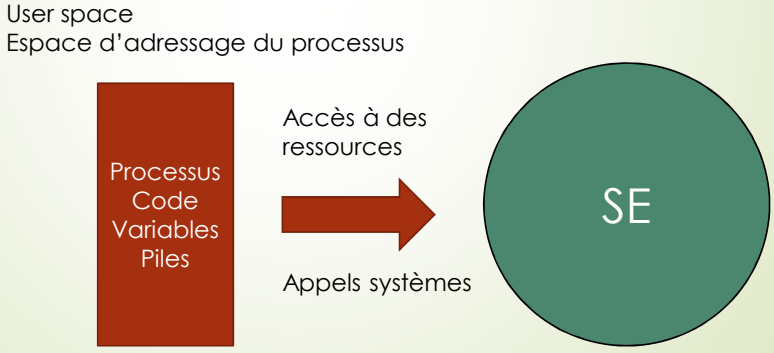
Conteneurs Linux: l'isolation et les namespaces

Anthony BUSSON

Exécution d'un processus

- Un processus s'exécute au sein de son espace d'adressage (en RAM).
- Il n'a pas le droit de lire ou d'écrire en dehors, il n'a pas directement accès aux ressources (fichiers, réseau, etc.) : en cela il est par nature isolé.
- L'interaction avec les ressources ou les autres processus se fait obligatoirement via des appels systèmes.
- Si il existe des services d'isolation, c'est forcément le système qui les implémentent.



Conteneurs: les outils linux

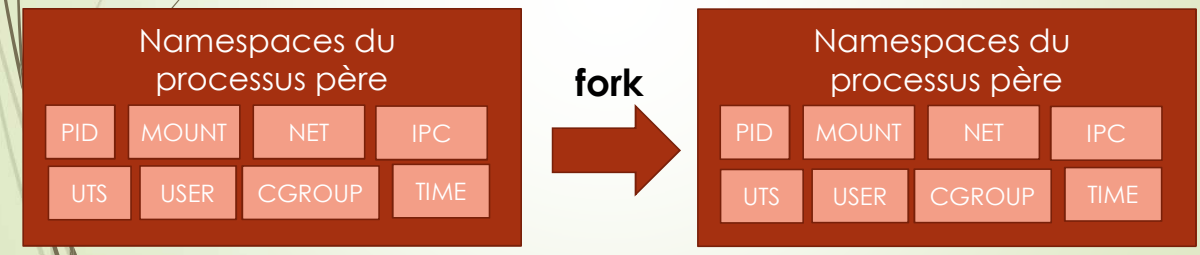
- Nous nous intéressons à l'isolation sous Linux (en cours sous windows)
- Les conteneurs sont basés sur des fonctionnalités du noyau Linux
  - namespace
  - cgroups
  - sécurité

namespace

- Depuis 2002
  - Linux 2.4.19 (dernière version 5.15.5)
- 8 différents namespace depuis la version 5.6 du noyau Linux
- Chaque type de namespace définit ce que le processus ou groupe de processus peut accéder et modifie le comportement des appels systèmes associés
  - mount: montage des périphériques / de systèmes de fichiers
  - process ID (pid)
  - network: interface, table de routage, ports, iptable, etc.
  - ipc (inter processes communication)
  - UTS (Unix Time Sharing): host and domain name
  - user ID: mapping des ID
  - time namespace: le temps peut être différent d'un namespace à l'autre (depuis 2020)
  - Control group namespace

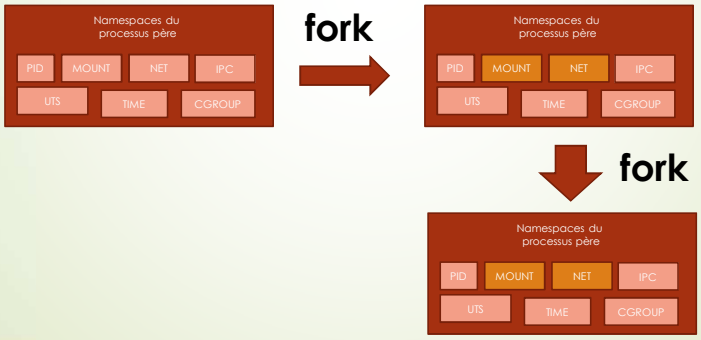
Principe

- Chaque processus appartient à chacun des namespace
- Par défaut, un processus hérite des namespace de son père
- Un processus peut créer son propre namespace
  - Par défaut, tous ses fils appartiendront au même namespace (sauf nouvel appel à unshare)



Principe

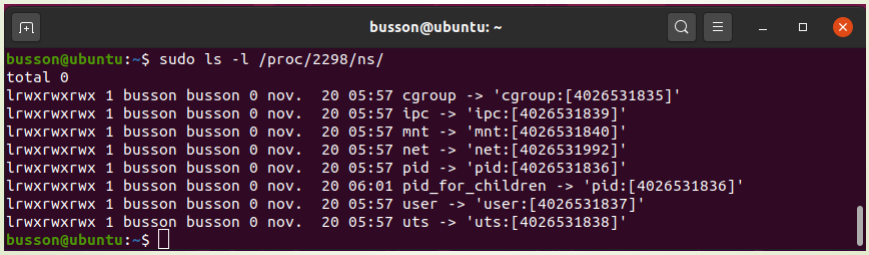
- Chaque processus appartient à chacun des namespace
- Par défaut, un processus hérite des namespace de son père
- Un processus peut créer son propre namespace
  - Par défaut, tous ses fils appartiendront au même namespace (sauf nouvel appel à unshare)



Identifiant d'un namespace

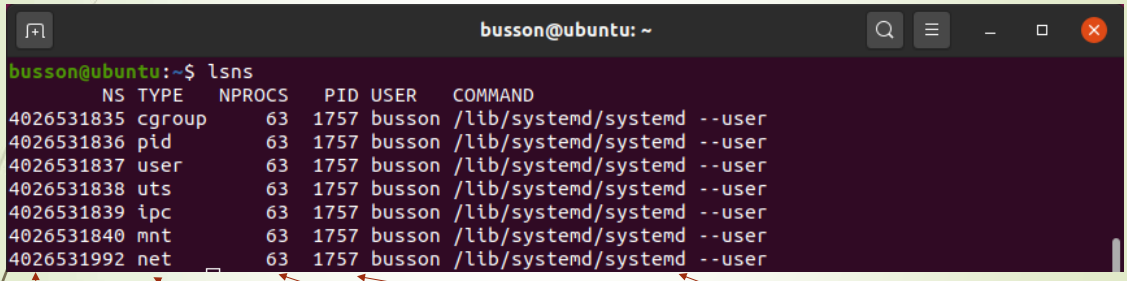
- Un namespace est identifié par un numéro d'inode.
- Tous les processus appartiennent à un namespace pour chaque catégorie (7 sous les versions actuelles de Linux/Ubuntu)
- Chaque processus a un lien symbolique vers chacun de ses namespaces
  - L'inode pointé a un numéro (d'inode) et un nom qui identifie le namespace
  - Tous les processus appartenant au même namespace ont ce même lien symbolique

/proc/<pid>/ns/<leLien>



lsns

- La commande lsns liste les namespace d'un processus



Identifiant du namespace: c'est un inode

Type de namespace Voir slide précédent

Nombre de processus dans ce namespace

Plus petit PID dans ce ns

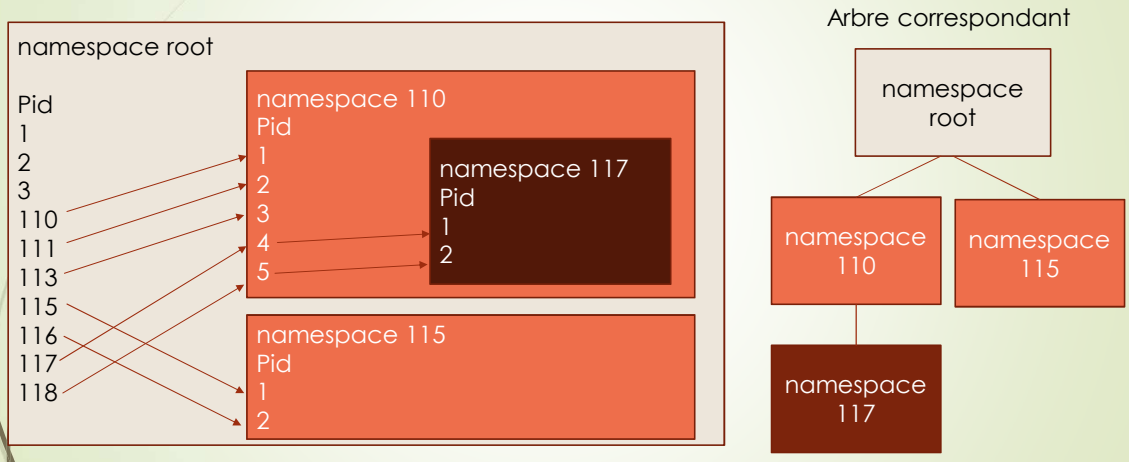
Commande (processus) pour lequel ces ns ont été affichés.

namespace pid

- Isolation des processus
  - Lorsqu'un processus est assigné a un nouveau namespace « pid »
    - Il récupère le pid 1 dans son propre namespace
    - Ses fils auront les pid 2, 3, etc.
  - Les processus ne peuvent interagir qu'avec les processus de leur propre namespace ou des namespace enfants
    - Interagir = utiliser des appels systèmes avec eux (signaux, etc.)

Namespace pid: détail (1)

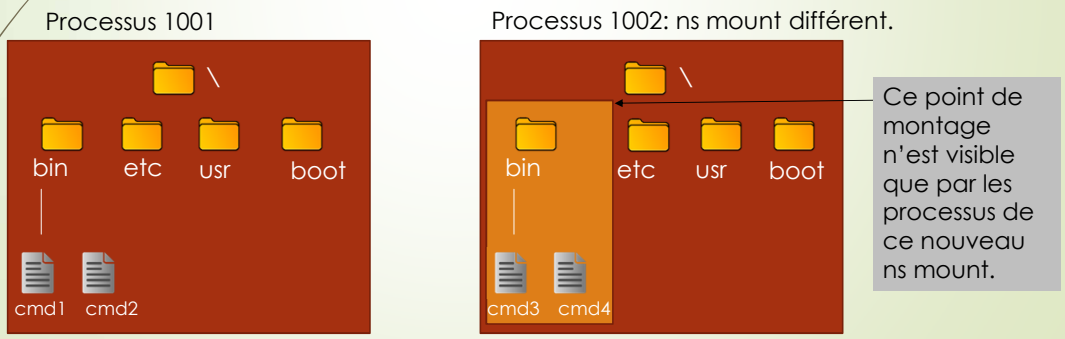
- Les namespace sont emboîtés (nested)



Le processus de pid 117 a 3 PID: un dans son propre namespace (1 ici), et un dans chacun des namespaces de niveau supérieur (4 et 117).

Namespace mount

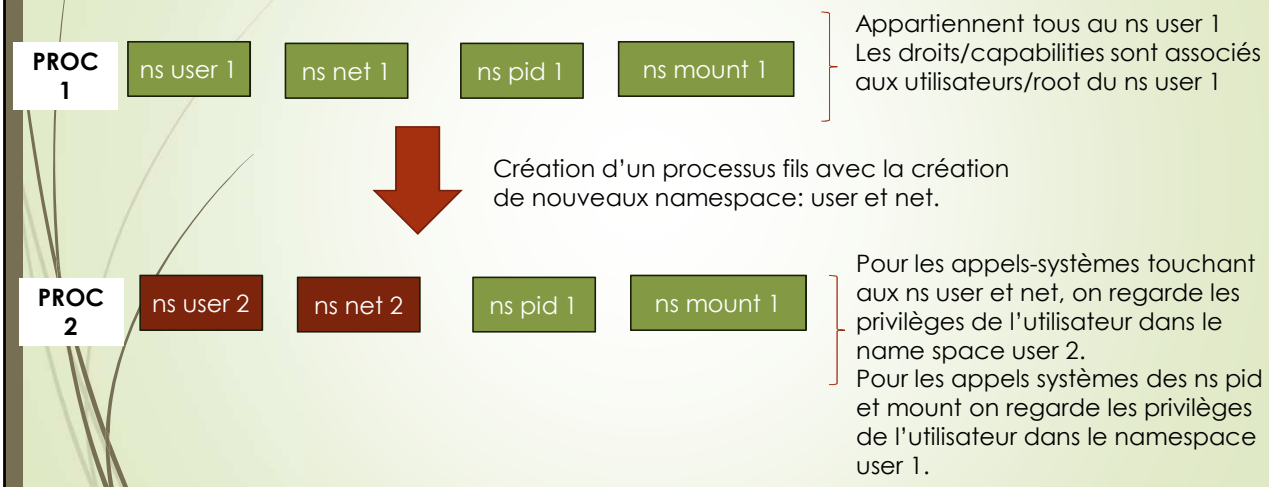
- Isole les points de montages pour chacun des « mount » namespace
- Chaque namespace « mount » a sa liste de points de montage
- Intérêt: monter des parties du système qui seront propre au processus de ce namespace (et pas visible par les autres processus).
  - On peut donc avoir des dossiers qui sont les mêmes que ceux du SE et d'autres qui sont locales au namespace (suivant ce qui a été monté).



Namespace UTS et user

- Namespace UTS: Permet de changer le hostname et domainname
  - Et aussi le NIS name (peu utilisé)
- Namespace user (exemple du « root mapping ») :
  - Associe l'uid 0 / gid 0 au processus qui créer le nouveau namespace
  - L'ensemble des namespace créer en même temps appartiennent à ce nouveau namespace user
  - Les capabilities de ces nouveaux namespace sont accessibles par cet utilisateur root (hostname, network, etc.).
  - Pour les autres namespaces, le root local n'a pas les droits.
  - Fonctionnement: mapping entre l'uid/gid du namespace et celui du host.

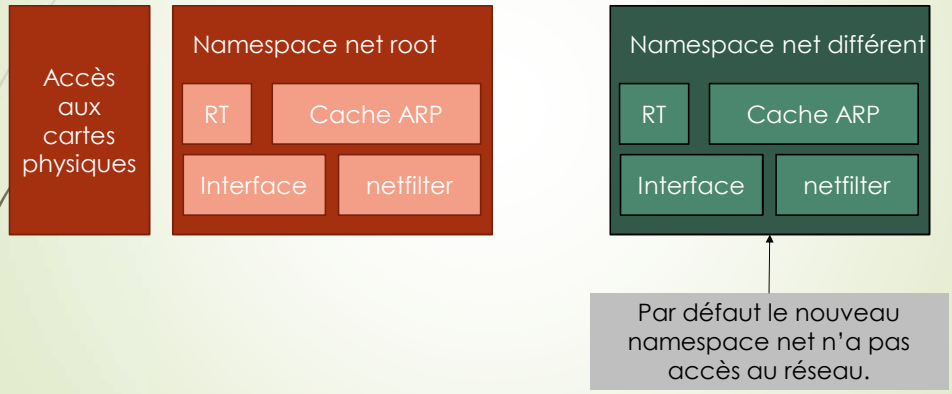
Exemple namespace user: le root mapping



Namespace net

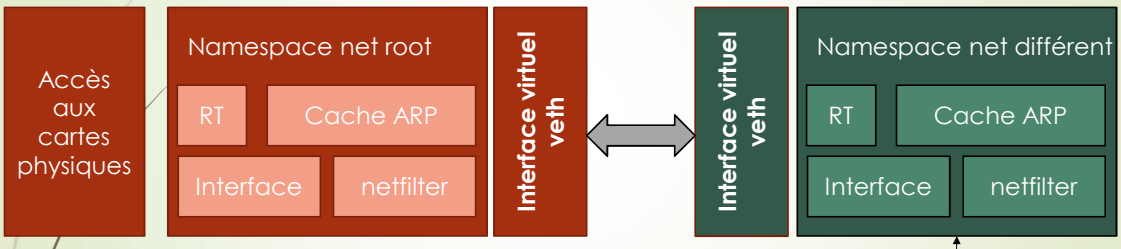
- Permet d'isoler les aspects réseau des processus.
- Pour chaque net namespace, les éléments suivants lui sont propres:
  - interface
  - Une table de routage, une table arp
  - Régles de filtrage (iptable)
  - Etc.
- Les commandes doivent s'appliquer au bon namespace

Namespace net



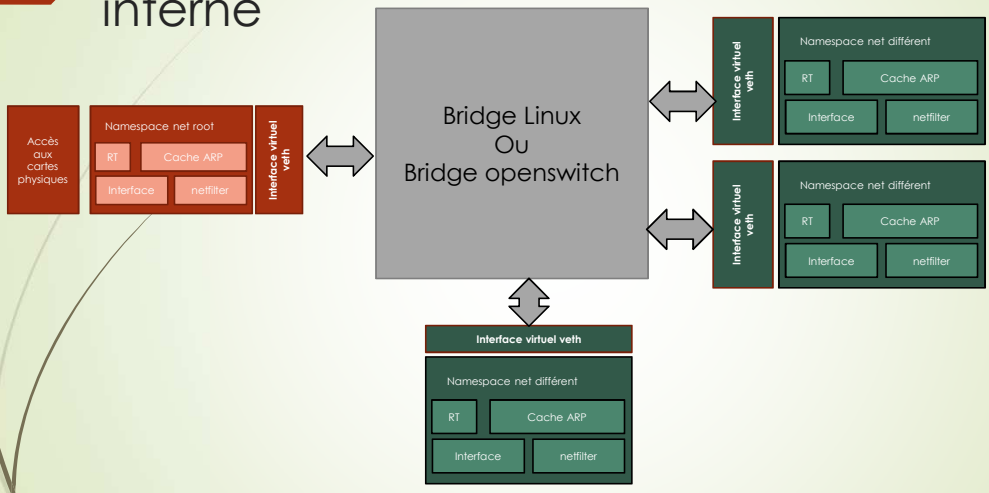
Par défaut le nouveau namespace net n'a pas accès au réseau.

Namespace net: se connecter au namespace root



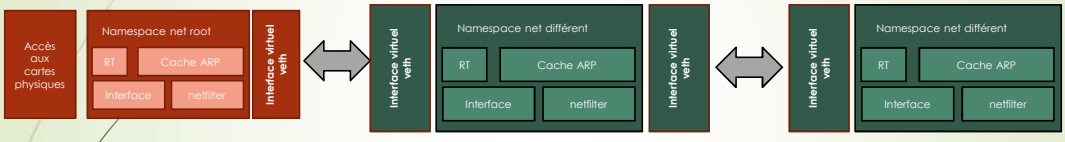
Par défaut le nouveau namespace net n'a pas accès au réseau.
Des interfaces virtuelles (dans chaque namespace) permettent d'interconnecter les namespace net entre eux et potentiellement d'accéder aux interfaces physiques via le namespace root.

Namespace net: utilisation d'un bridge interne



Possibilité de créer un ou des « bridges » internes au système. Ils seront utilisés pour interconnecter les namespace net.

Namespace net: topologie arbitraire



Il est possible de créer des topologies arbitraires avec un jeu d'interfaces virtuelles, éventuellement des bridges, et des routes adaptées.
Les namespaces peuvent alors avoir des rôles différents, de filtrages par exemple.

Namespace IPC

- Isolation des IPC system V
  - Sémaphores
  - Files de messages
  - Mémoire partagée

Les commandes

- lsns: affiche les ns d'un processus
- unshare:
  - associe un namespace à un processus existant
  - créer un processus avec un ou des namespaces différents
- nsenter:
  - permet de taper des commandes associés à certains namespace
    - Changer l'utilisateur
    - Monter des systèmes de fichiers ou périphériques
    - Changer le hotsname
    - Etc.

cgroups (control groups)

- Depuis 2008 (Linux version 2.6.24)
- Bien distinguer les cgroups (ci-dessous) et le namespace cgroup (slides suivants)
- Limite l'usage des ressources pour un ensemble de processus
  - Priorisation: limite l'usage CPU
  - Mémoire: alloue un volume maximum de mémoire au groupe
  - Comptabilité: compte le temps CPU
  - Isolation: un groupe de processus sera associé à des namespace (déjà fait).
- Actuellement: cgroup version 2

cgroups: fonctionnement

- Les cgroups sont créés/supprimés au travers de dossiers dans le répertoire suivant (pour le cpu ici / cela peut être memory aussi):

  /sys/fs/cgroup/cpu/
- Le cgroup initial incluant tous les processus est le cgroup « / » se trouvant à la racine de /sys/fs/cgroup/cpu
- La création d'un cgroup peut se faire avec mkdir:

  mkdir /sys/fs/cgroup/cpu/**myCgroup**
- Pour ajouter un processus à ce cgroup, il faut ajouter son pid dans le fichier

  /sys/fs/cgroup/cpu/myCgroup/cgroup.procs

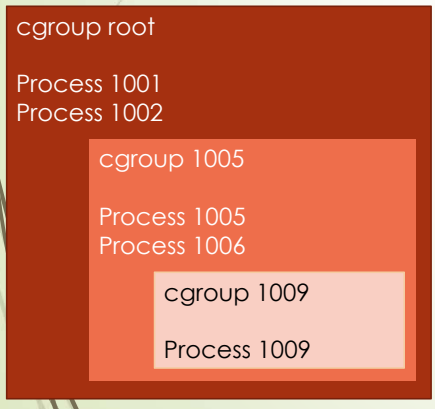
Les fils appartiennent aux cgroups de leur père.

cgroups: création des limites

- CPU:
  - On alloue des ressources sur une base globale de 1024
  - Par exemple: 512 limitera à 50% du CPU
  - Alloué dans le fichier /sys/fs/cgroup/cpu/myCgroup/cpu.shares
- Mémoire:
  - On alloue en nombre d'octets /sys/fs/cgroup/memory/myCgroup/memory.limit_in_bytes

cgroup hierarchy

- Cgroup est un moyen de limiter l'accès aux ressources CPU et mémoires
- Il est possible de créer une hierarchie entre les cgroups.

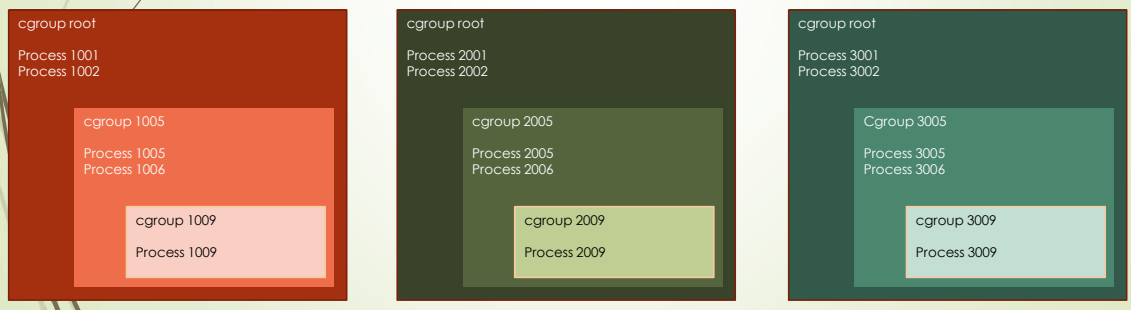


Hierarchie de processus:

- L'ensemble des proc se partagent les ressouurces du cgroup root.
- Les processus du cgroup 1005 se partagent des ressources plus contraignantes que le cgroup root, etc.

cgroup namespace

- Permet de créer des cgroup différents qui ne se voient pas (isolation)
- Racine différente pour chaque namespace



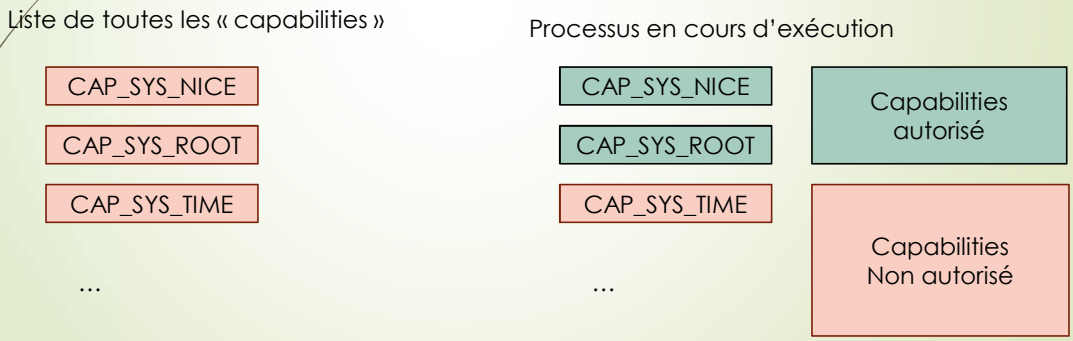
Sécurité: contrôler l'accès aux ressources (appels systèmes)

- Il existe 3 moyens de contrôler les appels systèmes qu'un processus a le droit d'appeler.

Namespace user Root mapping	Capabilities	seccomp
Root au sein de son Namespace.	Autoriser/interdire chaque tâche système.	Autoriser/interdire les tâches systèmes avec la granularité des appels systèmes.

Sécurité: capabilities

- Les capabilities sont associés à un processus donné.
- Ils regroupent des appels systèmes.
- Permet de limiter les droits root (pour certains processus)
- Permet d'augmenter les droits de certains processus (non root).
- Les capabilities sont modifiables mais héritées du père.

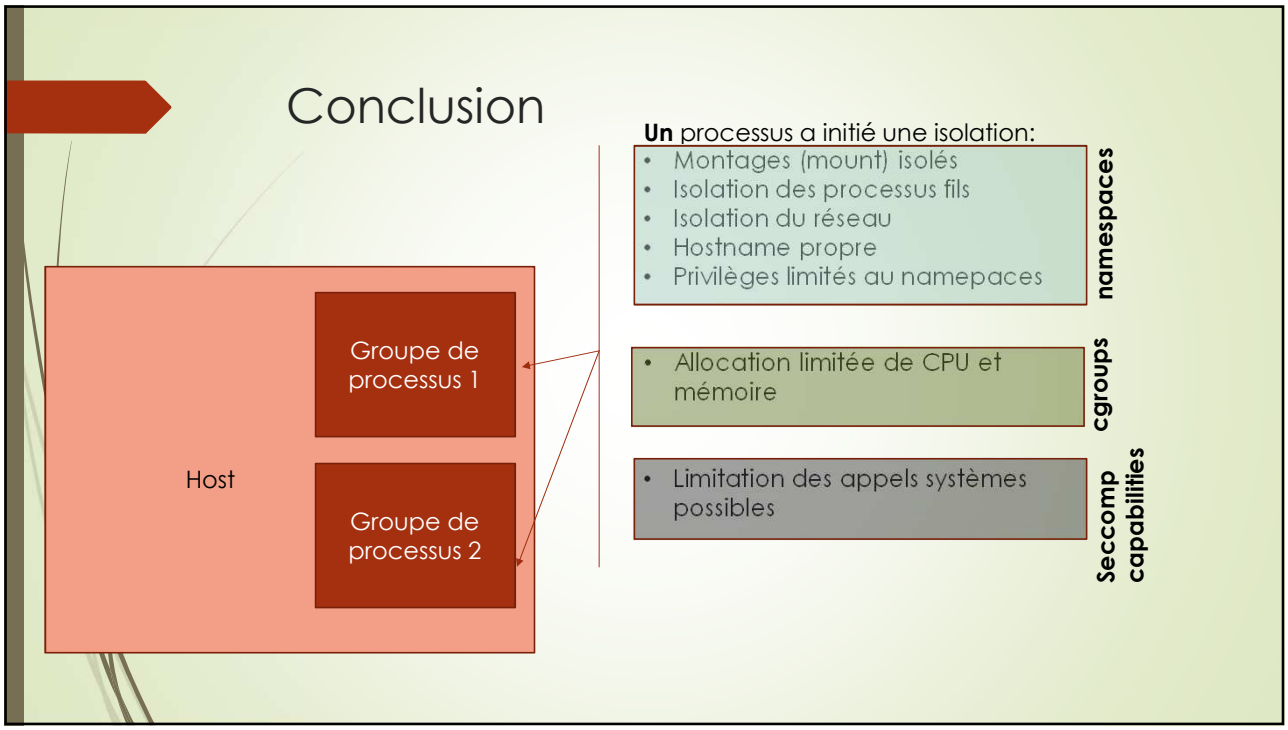


Sécurité

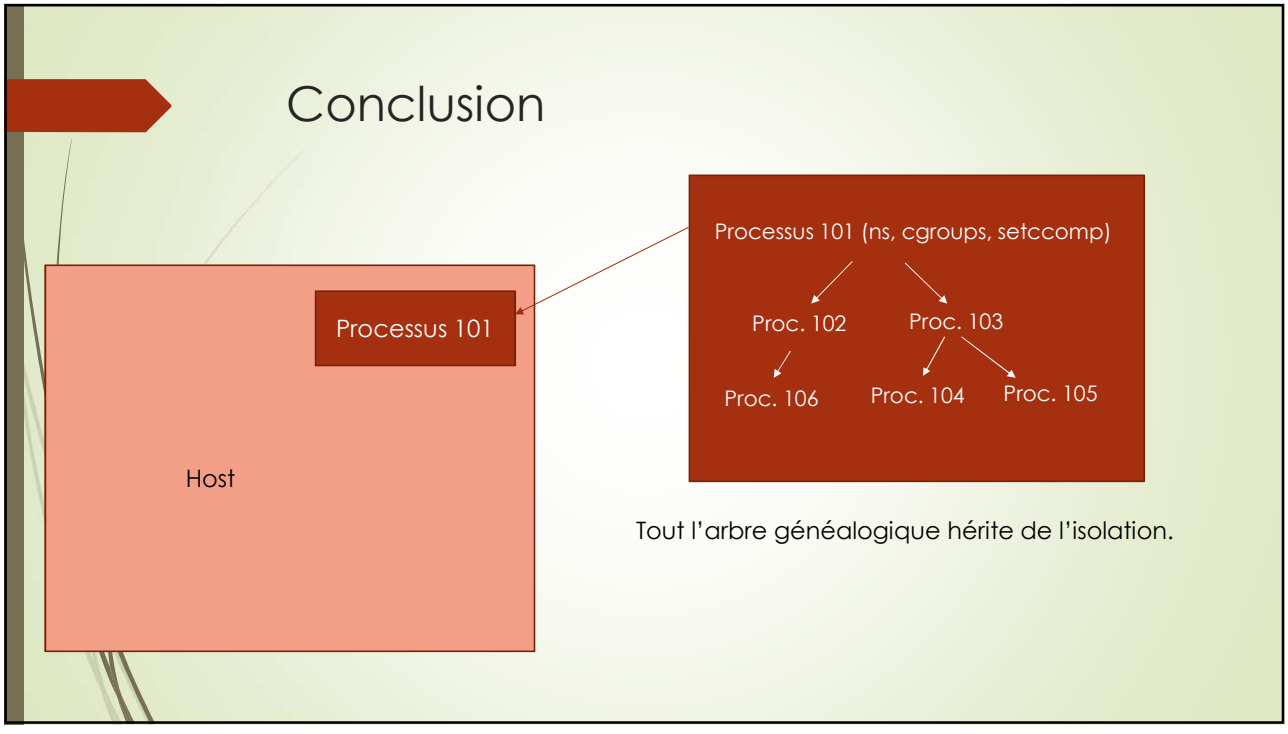
- L'appel système seccomp(2):
  - Filtre les appels systèmes qui peuvent être exécutés par un processus
  - Liste blanche
  - Liste noir
  - Une liste préétablie: read, write, exit, etc.

Les fils récupèrent les mêmes filtres que leur père.

Conclusion



Conclusion



Tout l'arbre généalogique hérite de l'isolation.