

# Sécurité des infrastructures de virtualisation

Timothée Ravier

🐦 @siosm

🌐 <https://tim.siosm.fr/cours>

5<sup>e</sup> année cycle ingénieur, filière STI  
Option Sécurité des Systèmes Ubiquitaires  
2018 - 2019

## Plan global

- 1 Infrastructures
- 2 Conteneurs
- 3 systemd
- 4 Gestionnaires de conteneurs
- 5 SELinux
- 6 Conteneurs et sécurité
- 7 Virtualisation et sécurité
- 8 Sécurité des infrastructures

1/281

## Plan

- 1 Infrastructures
- 2 Conteneurs
- 3 systemd
- 4 Gestionnaires de conteneurs
- 5 SELinux
- 6 Conteneurs et sécurité
- 7 Virtualisation et sécurité
- 8 Sécurité des infrastructures

2/281

## Infrastructures ?

### Infrastructure de type Cloud

Ensemble de **services** qui fournissent aux **utilisateurs** un **environnement** pour déployer des **applications** sous forme de **machines virtuelles, conteneurs** ou « **fonctions** ».

### Self service

Chaque utilisateur peut choisir ce dont il a besoin parmi les services proposés (stockage, réseau, base de données, gestion de charge, etc.).

### Responsabilités séparées

Les services fournis par l'infrastructure ne sont pas administrés par les utilisateurs de l'infrastructure.

3/281

## Services fournis par une infrastructure

- Compute :
  - Machines virtuelles
  - Conteneurs
  - Serverless (ou « Fonctions »)
- Stockage :
  - Stockage d'images disques (block storage)
  - Stockage d'images de conteneurs (container registry)
  - Stockage d'objets / blocs de données (object storage)
  - Stockage persistant (NFS, etc.)
- Réseau :
  - DNS
  - Load Balancing
  - Enregistrement et découverte de services
  - Routage

4 / 281

## Services fournis par une infrastructure

- Bases de données
- Gestion des secrets
- Gestion des identités
- Gestion de la configuration
- Intégration Continue (CI) :
  - Buildbot
  - Construction de conteneurs
  - Test automatique
- Supervision :
  - Métriques (performance)
  - Inventaire et audit
- Intégration avec du matériel (routeurs, stockage, etc.)
- Intégration avec les services de Cloud providers

5 / 281

## Exemples d'infrastructures

Cloud providers :

- Amazon Web Services,
- Google Cloud,
- IBM Cloud,
- Microsoft Azure,
- Alibaba Cloud,
- Yandex Cloud,
- etc.

6 / 281

## Mettre en place une infrastructure

Nombreux projets proposant un ensemble de logiciels pour mettre en place sa propre infrastructure, dans un environnement public (Cloud provider) ou privé (datacenter) :

- Infrastructure orientée virtualisation :
  - OpenStack;
  - oVirt;
  - Apache CloudStack, Eucalyptus, OpenNebula, etc.
- Infrastructure orientée conteneurs :
  - Kubernetes (et dérivés : OpenShift, etc.);
  - Hashicorp Nomad;
  - Mesos, Docker Compose, Cloud Foundry, etc.

7 / 281

# Sécurité des infrastructures de virtualisation

- Sécurité des technologies sur lesquelles se base l'infrastructure :
  - Sécurité des hôtes ;
  - Sécurité des machines virtuelles ;
  - Sécurité des conteneurs.
- Sécurité des services :
  - qui composent l'infrastructure ;
  - qui sont fournis aux utilisateurs par l'infrastructure ;
  - Voir dernier chapitre.

8/281

# Plan

- 1 Infrastructures
- 2 Conteneurs
  - Historique et concepts
  - Gestion des ressources (cgroups)
  - Isolation (namespaces)
  - Capabilities & seccomp
- 3 systemd
- 4 Gestionnaires de conteneurs
- 5 SELinux
- 6 Conteneurs et sécurité
- 7 Virtualisation et sécurité
- 8 Sécurité des infrastructures

9/281

# Plan

- 2 Conteneurs
  - Historique et concepts
  - Gestion des ressources (cgroups)
  - Isolation (namespaces)
  - Capabilities & seccomp

10/281

# Cas d'usage de la virtualisation

- Avantages :
  - Bonne isolation ;
  - Contrôle complet des services et interactions ;
  - Options possibles au niveau du partage du matériel ;
  - Gestion des ressources simple mais fixe.
- Désavantages :
  - Système d'exploitation complet à gérer ;
  - Consommation en ressources importante ;
  - Temps de démarrage ;
  - Duplication des fonctionnalités pour chaque instance : journalisation, contrôle d'accès, etc.

Conclusion : Rarement intéressant de lancer une VM classique pour chaque instance d'une application.

11/281

## Cas d'usage des conteneurs (sous Linux)

### Objectifs :

- Créer un ensemble de processus indépendant du reste du système ;
- Maîtriser la consommation en ressources système ;
- Isoler ces processus du reste du système ;
- Sans nécessairement utiliser la virtualisation ;
- Démarrage très rapide ;
- Durée de vie plus variable qu'une machine virtuelle.

12/281

## Cas d'usage des conteneurs (sous Linux)

### En pratique :

- Matériel : abstraction fournie par le noyau souvent suffisante (pas besoin d'émulation (QEMU), même architecture, etc.) ;
- Noyau : la version du noyau n'importe pas pour la plupart des applications ;
- Complexité moindre (moins d'éléments) pour debugger ;
- Partage des services système entre les conteneurs (journalisation, etc.) ;
- Temps de démarrage, taille et consommation en ressource moindre.

13/281

## Virtualisation ?

On parle de virtualisation légère, virtualisation au niveau du système d'exploitation ou de conteneurs par opposition à la virtualisation lourde/complète/totale d'un système d'exploitation.

14/281

## Historique

- chroot (1982) : exécuter un programme avec un / distinct ;
- FreeBSD Jails (2000) : ajoute des restrictions au niveau des interactions avec le noyau, le réseau et virtualise le super utilisateur ;
- Linux Vserver (2001) : concept de *context* et fonctionnalités similaire aux jails pour Linux. Patch noyau uniquement, support très limité.
- Solaris Zones (2004) : concept de *system call translation* ;
- OpenVZ (2005) : patch noyau Linux uniquement, support limité. Certaines fonctionnalités ont été intégrées upstream (CRIU).

15/281

## Attention !

Pas de concept de conteneur dans le noyau Linux !

Il faut combiner « manuellement » :

- la gestion des ressources et de l'accès à certains périphériques avec les cgroups ;
- l'isolation (modulable) avec les namespaces ;
- les restrictions de permissions avec les capabilities, les filtres seccomp et les Linux Security Modules (SELinux / AppArmor).

## 2 Conteneurs

- Historique et concepts
- Gestion des ressources (cgroups)
  - Gestion statique des ressources
  - Gestion dynamique : cgroups
- Isolation (namespaces)
- Capabilities & seccomp

- Limite les ressources disponibles pour l'ensemble des processus d'un utilisateur ou d'un groupe ;
- Affichage / manipulation des limites avec `ulimit` ;
- Deux types de limites : `hard` et `soft` ;
- Limite `soft` :
  - limite appliquée par le noyau ;
  - définissable entre 0 et `<limite hard>`.
- `CAP_SYS_RESOURCE` pour augmenter les limites `hard` ;
- Réduction des limites `hard` irréversibles pour `!root` ;
- Limites initiales appliquées lors du login par PAM, configurées dans `/etc/security/limits.conf`.

- |   |  |
|---|--|
| ■ cpu time (seconds) :<br>unlimited         | ■ locked-in-memory size<br>(kbytes) : 64 |
| ■ file size (blocks) : unlimited            | ■ address space (kbytes) :<br>unlimited  |
| ■ data seg size (kbytes) :<br>unlimited     | ■ file locks : unlimited                 |
| ■ stack size (kbytes) : 8192                | ■ pending signals : 22949                |
| ■ core file size (blocks) : 0               | ■ bytes in POSIX msg queues :<br>819200  |
| ■ resident set size (kbytes) :<br>unlimited | ■ max nice : 0                           |
| ■ processes : 1024                          | ■ max rt priority : 0                    |
| ■ file descriptors : 1024                   |  |

## Quota disque

- Limite les ressources disque par user et par group ;
- Limite `soft` & `hard` par système de fichier ;
- La limite `hard` ne peut pas être dépassée ;
- La limite `soft` peut être dépassée pour un temps donné ;
- Une fois ce temps dépassée, la limite `soft` devient `hard` ;
- L'utilisateur doit libérer de la place pour redescendre au dessous de la limite `soft` avant de pouvoir écrire à nouveau des données ;
- Quota disque par « projets » avec XFS [1].

20/281

## Espace disque réservé pour les processus privilégiés

- Il est possible de réserver de l'espace sur un système de fichier pour permettre aux processus privilégiés (syslog / journald) de continuer à fonctionner lorsque le disque est presque plein ;
- `$ tune2fs -m reserved-blocks-percentage`
- Valeur par défaut : 5 % de la taille du système de fichier.

21/281

## L'overcommit

- Chaque processus a l'impression d'être le seul à utiliser le matériel ;
- On lui propose donc plus de ressources que ce qui est réellement disponible sur la machine ;
- Exemple : l'espace d'adressage virtuel
  - $2^{32} \approx 4\text{Go}$  en 32bits. Pour l'instant  $2^{48}$  en 64 bits ;
  - `/proc/sys/vm/{overcommit_memory,overcommit_ratio}` ;
  - `Documentation/vm/overcommit-accounting`.
- En tenir compte lorsque l'on atteint les limites.

22/281

## Priorités, IO, et CPU

- `nice level` : Défini la priorité d'accès aux ressources CPU pour un processus ;
- `ionice` : Défini la priorité d'accès aux périphériques pour un processus ;
- `cpu affinity` : Oblige un processus à utiliser des cœurs définis.

23/281

## Limites du modèle statique

- Flexibilité très limitée ;
- Il faut gérer « à la main » les changements de limite : pas de répartition automatique entre les utilisateurs ou les processus ;
- Peu adapté au modèle où l'on va avoir beaucoup de machines virtuelles ou conteneurs utilisées par un seul utilisateur ;
- Ou à des services constitués de plusieurs processus que l'on souhaite gérer comme un tout.

24/281

## cgroups

- Introduit dans le noyau 2.6.24 par des ingénieurs travaillant chez Google ;
- Permet de regrouper des processus en leur associant un label ;
- Organisation hiérarchique ;
- Les processus fils héritent du cgroup du père ;
- Représenté par un système de fichier virtuel : `/sys/fs/cgroup/`.

25/281

## cgroups : controllers

- Configuration d'un ou plusieurs controllers pour chaque groupe de tâches ;
- Imposent des limites ou contraintes communes à l'ensemble des processus de chaque groupe.
- `cgroups(7)` contient la liste complète ainsi que les versions du noyau à partir desquelles chaque controller est disponible ;
- Liste des controllers disponibles sur un système : `/proc/cgroups`.

26/281

## cgroups : controllers disponibles

- `cpu` & `cpuacct` : mesure et limite l'utilisation CPU ;
- `cpusets` : répartition sur les CPUs / nœuds NUMA ;
- `memory` : mesure et limite l'utilisation de la mémoire ;
- `blkio` : mesure et limite l'utilisation des périphériques de type block ;
- `net_cls` & `net_prio` : classe et définit des priorités pour les paquets réseaux ;
- `devices` : restreint l'accès aux devices disponibles ;
- `freezer` : pause l'exécution des processus du groupe ;
- `perf_event` : groupes spécifiques au sous-système `perf` ;
- `rdma` : limite les accès RDMA et InfiniBand ;
- `hugetlb` : limite l'utilisation des Huge Page.
- `pids` : limite le nombre de PID disponibles.

27/281

## cgroups v1 & v2

Deux versions des cgroups :

- cgroups v1 :
  - chaque controller a sa propre hiérarchie ;
  - pose des problèmes de gestion, performance, complexité ;
  - diverses limites liées au design.
- cgroups v2 :
  - une seule hiérarchie commune à tous les controllers [3] ;
  - certains controllers pas (encore) disponibles ;
  - configuration différente pour certains controllers.

28/281

## cgroups v2 : controllers disponibles

- cpu : mesure et limite l'utilisation CPU ;
- cpusets : répartition sur les CPUs / nœuds NUMA ;
- memory : mesure et limite l'utilisation de la mémoire ;
- io : mesure et limite l'utilisation des périphériques de type block ;
- perf\_event : groupes spécifiques au sous-système *perf* ;
- rdma : limite les accès RDMA et InfiniBand ;
- pids : limite le nombre de PID disponibles.

29/281

## Plan

### 2 Conteneurs

- Historique et concepts
- Gestion des ressources (cgroups)
- Isolation (namespaces)
- Capabilities & seccomp

30/281

## Isolation avec les namespaces I

Principe général :

- Crée un ou plusieurs espace de nom (namespaces ou ns) pour séparer le contexte d'exécution d'un processus par rapport aux autres ;
- Isole les objets créés dans un namespace vis à vis des autres namespaces ;
- Lorsque tous les processus d'un namespace ont terminé leur exécution, les objets restant sont détruits ou renvoyés dans le namespace parent.

31/281



## Isolation avec les namespaces II

- Fonctionnalité ancienne (début dans la branche 2.4!) mais étendue récemment;
- Au départ ajouté comme *flags* à l'appel système `clone` :

```
int clone(int (*fn)(void *), void *child_stack,  
         int flags, void *arg, ...  
         /* pid_t *ptid, void *newtls, pid_t *ctid */ );
```

- `CLONE_NEWNET`, `CLONE_NEWNS`, `CLONE_NEWPID`, `CLONE_NEWUTS`, `CLONE_NEWIPC`, `CLONE_NEWUSER`, `CLONE_NEWCGROUP`...
- Nouveaux appels systèmes pour modifier le processus courant :

```
int unshare(int flags);  
int setns(int fd, int nstype);
```

- Voir *namespaces(7)*, *unshare(2)*, *clone(2)*, *setns(2)*.

32/281

## CLONE\_NEWNS

Crée un namespace pour les points de montage :

- Présent depuis le noyau 2.4.19 (2002);
- Permet au fils de monter/démonter des systèmes de fichier sans impacter le père;
- Nécessite `CAP_SYS_ADMIN`.

33/281

## CLONE\_NEWUTS

Crée un namespace UTS :

- Présent depuis le noyau 2.6.19 (2006);
- Contient le domain name et le host name;
- Permet au fils de changer ces informations sans impacter le père;
- Nécessite `CAP_SYS_ADMIN`.

34/281

## CLONE\_NEWIPC

Crée un namespace pour les IPC :

- Introduit dans le noyau 2.6.19, complété dans le 2.6.30 (2009);
- Vue isolée des IPC System V : seuls les processus de ce namespace peuvent accéder à ces objets;
- Destruction des objets à la fin du namespace;
- Nécessite `CAP_SYS_ADMIN`.

35/281

## CLONE\_NEWNET

Crée un namespace pour les interface réseaux :

- Introduit dans le noyau 2.6.24, complété dans le 2.6.29 (2009);
- Le fils dispose de sa propre stack réseau : interfaces, stacks IPv4/IPv6, tables de routage IP, règles de parefeu, sockets...
- Une interface physique ne peut être que dans un seul namespace à la fois : rattachée au namespace initial à la destruction;
- Il faut donc créer des interfaces virtuelles pour pouvoir les utiliser dans les namespaces;
- Nécessite `CAP_SYS_ADMIN`.

36/281

## CLONE\_NEWPID

Crée un namespace pour les PID :

- Présent depuis le noyau 2.6.24 (2008);
- Les PIDs recommencent à 1 dans ce namespace;
- Les processus du namespace parent voient les processus du namespace fils avec des PIDs différents;
- Les processus dans un namespace ne peuvent voir que ceux du même namespace et des namespaces créés par des processus de leur namespace;
- Nécessite `CAP_SYS_ADMIN`.

37/281

## CLONE\_NEWUSER

Crée un namespace pour les UID/GID :

- Introduit dans le noyau 2.6.24, finalisé dans le 3.8 (2013);
- Associe un ensemble d'UID/GID dans le nouveau namespace au couple UID/GID initial;
- `root` dans un user namespace sans l'être en dehors;
- Le noyau s'assure qu'il n'y a pas d'élévation de privilèges par rapport au couple UID/GID initial;
- Nécessite la mise en place d'une correspondance entre les UIDs à l'intérieur et à l'extérieur du user namespace;
- Mode *Unprivileged* disponible depuis le 3.8. Sinon `CAP_SYS_ADMIN`, `CAP_SETUID` and `CAP_SETGID`. Comportement variable suivant les distributions.

38/281

## CLONE\_NEWCGROUP

Crée un namespace pour les hiérarchies cgroups :

- Introduit dans le noyau 4.6 (2016);
- Abstraction des hiérarchies cgroups pour les conteneurs;
- Non privilégié.

39/281

## Outils en ligne de commande

- `unshare` : lancer un programme en utilisant de nouveaux namespaces ;
- `nsenter` : lancer un programme avec les namespaces d'un programme existant ;
- Visualiser les namespaces associés à un programme :

```
$ ls -l /proc/1006/ns/*
... /proc/1006/ns/cgroup -> 'cgroup:[4026531835]'
... /proc/1006/ns/ipc -> 'ipc:[4026531839]'
... /proc/1006/ns/mnt -> 'mnt:[4026531840]'
... /proc/1006/ns/net -> 'net:[4026531993]'
... /proc/1006/ns/pid -> 'pid:[4026531836]'
... /proc/1006/ns/pid_for_children -> 'pid:[4026531836]'
... /proc/1006/ns/user -> 'user:[4026531837]'
... /proc/1006/ns/uts -> 'uts:[4026531838]'
```

40/281

## Plan

### 2 Conteneurs

- Historique et concepts
- Gestion des ressources (cgroups)
- Isolation (namespaces)

### ■ Capabilities & seccomp

- Capabilities
- seccomp

41/281

## Capabilities I

- L'utilisateur `root` est tout puissant sur un système Linux ;
- Tous les autres utilisateurs sont non privilégiés par défaut ;
- Les *capabilities* Linux : donner une partie des privilèges de `root` à des programmes non privilégiés ;
- Exemple :
  - `CAP_NET_BIND_SERVICE` : Bind un socket sur un port < 1024 ;
  - `CAP_NET_ADMIN` : Configurer les interfaces réseaux ;
  - `CAP_SETUID` : Définir arbitrairement les UIDs d'un processus ;
  - `CAP_SYS_ADMIN` : Effectuer un grand nombre d'opération privilégiées ;
  - Voir **capabilities(7)** pour la liste complète.
- Peuvent aussi être associées à des exécutables pour être ainsi transmises aux processus résultant.

42/281

## Capabilities II

- Comme pour les UID/GID, chaque *thread* possède son propre ensemble de *capabilities* ;
- Leur état est stocké dans plusieurs variables (nommés *sets*) :
  - Effective : le set utilisé pour les contrôles dans le noyau ;
  - Permitted : limite actuelle des *capabilities* disponibles ;
  - Inheritable : préservées lors d'un appel à `execve` et conservées si le fichier exécuté les possède ;
  - Ambient : préservées lors d'un appel à `execve` dans tous les cas.
- Capabilities Bounding Set : limite définitive des *capabilities* disponibles pour un *thread* ;
- Ensemble de règles complexes pour déterminer quelles *capabilities* sont conservées / perdues / transmises lors des appels à `execve`, `setuid`, ...
- Voir **capabilities(7)**.

43/281

- De nombreuses *capabilities* sont en pratique équivalente à root :
  - CAP\_SYS\_ADMIN : the new root [41];
  - False Boundaries and Arbitrary Code Execution [40].

- seccomp : mode strict qui n'autorise un processus à ne faire usage qu'aux appels système `read`, `write` et `_exit`;
- seccomp-bpf : extension du modèle initial qui filtre les appels système disponibles pour un processus;
- Filtre décrit sous forme de programme BPF (*Berkeley Packet Filter*);
- Si le processus utilise un appel système filtré, le noyau peut retourner un code d'erreur, d'envoyer un signal ou tuer le processus;
- En pratique : utilisation de la `libseccomp`, des options de `systemd` ou du gestionnaire de conteneurs;
- Possibilité de créer des blacklists / whitelists.

- Sécurité
- 1 Infrastructures
- 2 Conteneurs
- 3 systemd
  - Historique
  - Fonctionnement
  - journald
  - logind, networkd, timesyncd, resolved, etc.
- 4 Gestionnaires de conteneurs
- 5 SELinux
- 6 Conteneurs et sécurité
- 7 Virtualisation et sécurité
- 8 Sécurité des infrastructures

- 3 systemd
  - Historique
  - Fonctionnement
  - journald
  - logind, networkd, timesyncd, resolved, etc.
  - Sécurité

## Histoire : SystemV et sysinit

- Runlevels (`/etc/inittab`);
- Script shell pour chaque service (`/etc/init.d/nginx`);
- Beaucoup de répétitions, peu de mutualisation, beaucoup de hacks;
- Démarrage séquentiel, arrêt séquentiel;
- Chaque distribution a ses propres scripts pour chaque service;

48/281

## Histoire : SystemV et sysinit

- Difficile de maintenir des modifications dans la durée;
- Aucune information sur l'état du système;
- Pas de surveillance de l'état du système;
- Difficile de suivre les processus dans la durée (`/var/run/nginx.pid?`);
- Le processus PID 1 ne sert plus à rien une fois le système démarré.

49/281

## Histoire : Ubuntu : Upstart

- Démarrage des service à la suite d'événements;
- La carte bluetooth est disponible  $\Rightarrow$  démarre le service bluetooth;
- Le réseau est disponible  $\Rightarrow$  montage des disques distants;
- Difficile d'empêcher un service de démarrer automatiquement;
- Meilleur que SysV, mais pas assez puissant.

50/281

## Histoire : Mac OS X : launchd

- Introduit avec Mac OS X 10.4;
- Démarre le système et gère les services;
- Démarrage des services en parallèle et à la demande;
- Démon launchd (PID 1) et « télécommande » `launchctl` pour gérer le les services;
- Gestion des daemons par utilisateur possible;
- Configuration en XML.

51/281

## Histoire : et les autres...

D'autres systèmes sont présentés dans l'article [12]. Ils reposent généralement sur un langage de script :

- OpenRC (Gentoo);
- rcNG (NetBSD, FreeBSD et DragonFlyBSD);
- runit;
- Service Management Facility (SMF, OpenIndiana).

52/281

## systemd

- Première version annoncée le 30 mars 2010;
- Développeurs principaux : Lennart Poettering, Kay Sievers, Harald Hoyer, Daniel Mack, Tom Gundersen, David Herrmann;
- Dernière version : 240, sortie le 21 décembre 2018;
- Adopté comme `init` par défaut par pratiquement toutes les distributions majeures (Fedora, Mageia, openSUSE, Arch Linux, CoreOS, RHEL, SUSE Linux Enterprise Server, Ubuntu, Debian).

53/281

## systemd

- Au début : juste un remplaçant d'`init`;
- Désormais : ensemble de blocs de base pour construire un système Linux.
- Beaucoup de rumeurs et fausses informations circulent à propos de systemd [13];
- Adapté à tous les environnements : Distributions embarquées, voitures, téléphones, tablettes, supercalculateurs, desktops...

54/281

## Plan

- 3 systemd
  - Historique
  - Fonctionnement
  - journald
  - logind, networkd, timesyncd, resolved, etc.
  - Sécurité

55/281

## Fonctionnalités

- Gestionnaire des **services** et du **système** : vue globale ;
- Contrôle des « **units** » plutôt que des démons ou processus ;
- Gestion des dépendances entre les units ;
- Suivi des processus qui composent chaque unit :
  - Correspondance unit  $\Leftrightarrow$  processus ;
  - Arrêt **propre** et **maîtrisé** des units.
- Temps de démarrage minimal ;
- Debuggable : Reproductibilité, récupération de tous les logs du système.
- Meilleure interface utilisateur (`systemctl`) ;
- « Compatible » avec l'existant.

56/281

## Démons principaux

### Système

- `systemd` (PID 1), `udev`, `journald`, `logind`

### Utilisateur, lancé à la demande

- `systemd --user`

### Système, lancés à la demande

- `localed`, `hostnamed`, `timedated`

### Système, optionnels, lancés à la demande

- `timesyncd`, `machined`, `networkd`, `resolved`, `socket-proxyd`

57/281

## Démons principaux

### Système

- **`systemd` (PID 1), `udev`, `journald`, `logind`**

### Utilisateur, lancé à la demande

- `systemd --user`

### Système, lancés à la demande

- `localed`, `hostnamed`, `timedated`

### Système, optionnels, lancés à la demande

- `timesyncd`, `machined`, `networkd`, `resolved`, `socket-proxyd`

58/281

## Init : PID 1

- « Premier » processus lancé par le noyau ;
- Succède dans le cas général à `initrd` ;
- Les partitions `/` et `/usr` ont été montées par `initrd` ;
- Chargé de démarrer le reste du système ;
- Parent de tous les processus orphelins.

59/281

## Init : **systemd** PID 1

- Spécialiste du **lancement de processus** :
  - **Quoi ?** service nginx, mysql...
  - **Comment ?** UID, GID, CWD, capabilities, env...
  - **Quand ?** au démarrage, gestion des dépendances, activation suite à un timer, une socket, un message D-Bus...
  - **Qui ?** Logs.
- et de leur **suivi** :
  - **Qui ?** Quels processus font parti de mon service ?
  - **Status** : watchdogs, socket de notification ;
  - **Redémarrage** : en cas d'échec, à la réception d'un SIGSEGV ;
  - **Arrêt propre** : aucun processus oublié.

60/281

## cgroups

- systemd place les **processus** de chaque **unit** dans un **cgroup distinct** ;
- systemd « **seul** » **gestionnaire** des cgroups ;
- Changement de granularité : processus  $\Rightarrow$  service (cgroup) ;
- Réponse facile aux questions :
  - A quel service appartient ce processus ?
  - Quels processus font partie de ce service ?

61/281

## cgroups : **systemd-cgls**

```
Control group /:  
- .slice  
  - init.scope  
    - 1 /usr/lib/systemd/systemd --switched-root --system --deserialize 24  
  - machine.slice  
    - machine-qemu\x2dfedora22.scope  
      - 11026 /usr/sbin/qemu-system-x86_64 -name fedora22 -S -machine pc-i440fx-2.3,accel=kvm...  
  - system.slice  
    - dbus.service  
      - 569 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation  
    - systemd-journald.service  
      - 309 /usr/lib/systemd/systemd-journald  
    - systemd-udevd.service  
      - 339 /usr/lib/systemd/systemd-udevd  
  - user.slice  
    - user-1000.slice  
      - user@1000.service  
        - pulseaudio.service  
          - 1446 /usr/bin/pulseaudio --daemonize=no  
          - 1466 /usr/lib/pulse/gconf-helper  
        - gpg-agent.service  
          - 1304 /usr/bin/gpg-agent --daemon --enable-ssh-support --pinentry-program /usr/bin/pinentry-qt  
        - init.scope  
          - 1289 /usr/lib/systemd/systemd --user  
          - 1290 (sd-pam)  
      - session-c2.scope  
        - 1411 kwin_x11  
        - 6255 /usr/bin/tmux  
        - 19664 /usr/bin/dolphin  
        - 24500 /usr/bin/thunderbird  
        - 27086 /usr/bin/zsh
```

62/281

## Units **systemd**

### 11 types d'units :

- **nginx.service**
- **multi-user.target**
- **sshd.socket**
- **systemd-tmpfiles-clean.timer**
  
- **-mount**, **home.mount**
- **acpid.path**
- **system.slice**, **user-1000.slice**
- **session-1.scope**
  
- **sys-devices-virtual-block-dm\x2d0.device**
- **proc-sys-fs-binfmt\_misc.automount**
- **dev-mapper-system\x2dswap.swap**

63/281



## Units : Emplacements

- (/usr)/lib/systemd/system : Fournies par la distribution (lecture seule);
  - /run/systemd/system : Générées à l'exécution, non persistantes;
  - /etc/systemd/system : Créées et modifiées par l'administrateur.
- Ordre de priorité : /etc/ → /run/ → /usr/lib/
- Liste des units :
- installées : `systemctl list-unit-files`
  - chargées : `systemctl list-units`

64/281

## Units : **httpd.service**

```
# systemctl cat httpd.service

1 [Unit]
2 Description=Apache Web Server
3 After=network.target remote-fs.target nss-lookup.target
4
5 [Service]
6 Type=forking
7 PIDFile=/run/httpd/httpd.pid
8
9 ExecStart=/usr/bin/apachectl start
10 ExecStop=/usr/bin/apachectl graceful-stop
11 ExecReload=/usr/bin/apachectl graceful
12
13 PrivateTmp=true
14 LimitNOFILE=infinity
15
16 [Install]
17 WantedBy=multi-user.target
```

65/281

## Units : **httpd.service**

```
# systemctl show -p Description httpd.service

1 [Unit]
2 Description=Apache Web Server
```

66/281

## Units : **httpd.service**

```
# systemctl start httpd.service

1 [Unit]
2 Description=Apache Web Server
3
4 [Service]
5 Type=forking
6 PIDFile=/run/httpd/httpd.pid
7
8 ExecStart=/usr/bin/apachectl start
```

67/281

## Units : **httpd.service**

```
# systemctl stop httpd.service

1 [Unit]
2 Description=Apache Web Server
3
4 [Service]
5 Type=forking
6 PIDFile=/run/httpd/httpd.pid
7
8 ExecStart=/usr/bin/apachectl start
9 ExecStop=/usr/bin/apachectl graceful-stop

# systemctl kill -signal=SIGTERM httpd.service
```

68/281

## Units : **httpd.service**

```
# systemctl reload httpd.service

1 [Unit]
2 Description=Apache Web Server
3
4 [Service]
5 Type=forking
6 PIDFile=/run/httpd/httpd.pid
7
8 ExecStart=/usr/bin/apachectl start
9 ExecStop=/usr/bin/apachectl graceful-stop
10 ExecReload=/usr/bin/apachectl graceful
```

69/281

## Pages de manuel correspondantes

- Toutes les pages de man sont listées dans **systemd.index(7)** ;
- Toutes les options d'unité sont listées dans **systemd.directives(7)** ;
- Options génériques pour toutes les unités dans **systemd.unit(5)** ;
- Options spécifiques aux services dans **systemd.service(5)**.

70/281

## Exemple : **systemctl status**

```
$ sudo systemctl status crond.service
● crond.service - Periodic Command Scheduler
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; vendor preset: disabled)
   Active: active (running) since mar. 2016-01-05 03:01:07 CET; 1 weeks 1 days ago
     Main PID: 575 (crond)
       Tasks: 1 (limit: 512)
      CGroup: /system.slice/crond.service
              └─575 /usr/bin/crond -n

janv. 12 17:48:01 hydra anacron[23364]: Job `cron.weekly' terminated
janv. 12 18:01:01 hydra CROND[23982]: (root) CMD (run-parts /etc/cron.hourly)
janv. 12 19:01:01 hydra CROND[26767]: (root) CMD (run-parts /etc/cron.hourly)
janv. 13 13:01:01 hydra CROND[14366]: (root) CMD (run-parts /etc/cron.hourly)
janv. 13 13:01:01 hydra anacron[14371]: Anacron started on 2016-01-13
janv. 13 13:01:01 hydra anacron[14371]: Will run job `cron.daily' in 10 min.
janv. 13 13:01:01 hydra anacron[14371]: Jobs will be executed sequentially
janv. 13 13:40:53 hydra anacron[14371]: Job `cron.daily' started
janv. 13 14:01:01 hydra CROND[15078]: (root) CMD (run-parts /etc/cron.hourly)
janv. 13 15:01:01 hydra CROND[8012]: (root) CMD (run-parts /etc/cron.hourly)
```

71/281

## Gestion des dépendances

- Gestion implicite :
  - Activation via une socket TCP ou UNIX;
  - Activation via D-Bus.
- Gestion explicite :
  - Dépendance : **Requires=** / **Wants=** :
    - `systemctl show -p Requires nginx`
    - `systemctl show -p Wants nginx`
    - `systemctl list-dependencies nginx`
    - `systemctl list-dependencies --reverse nginx`
  - Ordre : **After=** / **Before=** :
    - `systemctl show -p After nginx`
    - `systemctl show -p Before nginx`
    - `systemctl list-dependencies --after nginx`
    - `systemctl list-dependencies --before nginx`

72/281

## Comment ordonner le lancement d'un service ?

Démarrer nginx après mysql et seulement si son lancement a réussi :

```
$ cat /etc/systemd/system/nginx.service.d/after-mysql.conf
```

```
1 [Unit]
2 Requires=mysql.service
3 After=mysql.service
```

73/281

## Units : **target**

- Remplaçant des runlevels SysVinit;
- Points de synchronisation;
- Regroupe plusieurs units à l'aide des dépendances;
- Plus flexible (pas de limite en quantité et simultanéité).

74/281

## Units : **multi-user.target**

```
1 [Unit]
2 Description=Multi-User System
3 Documentation=man:systemd.special(7)
4
5 Requires=basic.target
6 Conflicts=rescue.service rescue.target
7 After=basic.target rescue.service rescue.target
8
9 AllowIsolate=yes
```

75/281

## Units : `httpd.service`

```
1 [Unit]
2 Description=Apache Web Server
3 After=network.target remote-fs.target nss-lookup.target
4
5 [Service]
6 Type=forking
7 PIDFile=/run/httpd/httpd.pid
8
9 ExecStart=/usr/bin/apachectl start
10 ExecStop=/usr/bin/apachectl graceful-stop
11 ExecReload=/usr/bin/apachectl graceful
12
13 [Install]
14 WantedBy=multi-user.target
```

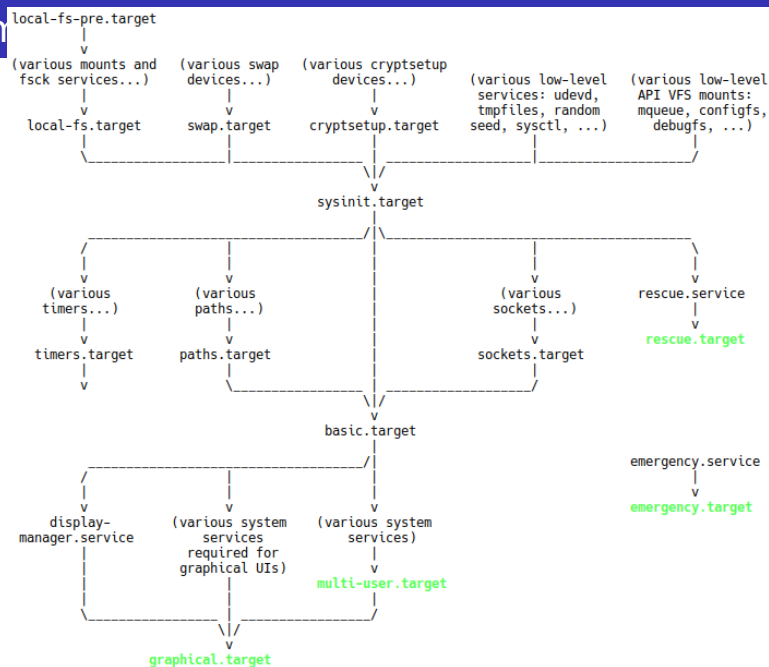
76/281

## Démarrage du système

- Vers quelle target pointe `default.target` ?
- `Boot` ⇔ `systemctl start default.target`
- Résolution des dépendances ;
- Démarrage des units ;
- Apparition des devices.

77/281

## Dén



78/281

## Usage : Démarrage au boot

```
# systemctl enable httpd.service
ln -s /usr/lib/systemd/system/httpd.service \
    /etc/systemd/system/multi-user.target.wants/
```

```
# systemctl disable httpd.service
rm /etc/.../multi-user.target.wants/httpd.service
```

```
# systemctl mask httpd.service
ln -s /dev/null /etc/systemd/system/httpd.service
```

79/281

## Usage : Démarrage lorsque le réseau est OK

Exemple sous Debian :

- Target : `network-online.target` ;
- Pas de dépendance par défaut ;
- Choisir ce qui permet d'atteindre l'état « réseau OK » ;
- Les interfaces définies dans le fichier `/etc/network/interfaces` sont configurées :

```
$ ln -s /lib/systemd/system/networking.service  
/etc/systemd/system/network-online.target.requires/
```

- Service devant attendre que le réseau soit OK :

```
$ cat /etc/systemd/system/foo.service.d/bar.conf
```

```
1 [Unit]  
2 Requires=network-online.target  
3 After=network-online.target
```

80/281

## Interaction **udev** / **systemd**

- Entrée dans `/etc/fstab` :

```
1 /dev/sda3 /home ext4 rw,nodev,nosuid 0 2
```

- Exemple :

- ① `systemd-fstab-generator` ⇒ `home.mount`
- ① Attends la découverte du périphérique de type block : `dev-sda3.device` ;
- ② Vérification du système de fichier : `systemd-fsck@dev-sda.service` ;
- ③ Monte le système de fichier.

81/281

## Débugger **systemd** ?

Un démon ne marche pas ?

- `# systemctl --state failed`
- `# systemctl status foo.service`
- `# systemctl show foo.service`
- `# journalctl -e`
- `# journalctl -e -u foo.service`
- `# journalctl -e /usr/bin/foo`
- `# journalctl -e _PID=123456`

82/281

## Débugger **systemd** ?

Erreurs au démarrage ?

- Erreur courante : configuration `/etc/fstab` incorrecte ;
- Options noyau :
  - `systemd.unit=rescue.target`
  - `systemd.unit=emergency.target`
  - `systemd.log_level=debug systemd.log_target=kmsg log_buf_len=1M`
  - `systemd.debug-shell (tty9)`
  - `systemd.confirm_spawn=true`
- `# systemctl list-jobs`

83/281

## Débugger systemd ?

Récupérer les logs des précédents démarrage :

- Dans `/etc/systemd/journald.conf` :

```
1 Storage=persistent
```

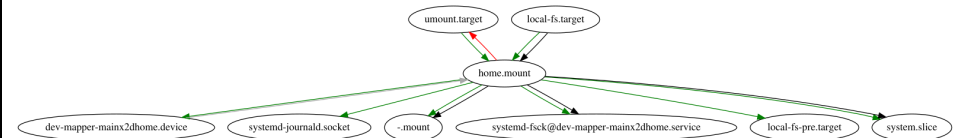
- # `journalctl -b -1`

84 / 281

## Débugger systemd ?

Problème de dépendance ?

- # `systemctl list-dependencies`
- # `systemd-analyze dot home.mount | dot -Tsvg > plt.svg`
- \$ `firefox plt.svg`

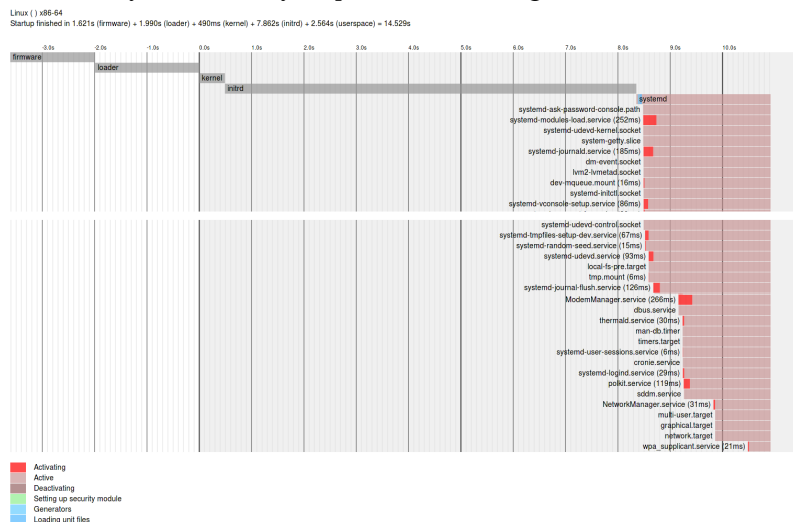


85 / 281

## Débugger systemd ?

Problème de performance ?

- # `systemd-analyze plot > boot.svg`



86 / 281

## Débugger systemd ?

Problème de performance ?

\$ `systemd-analyze critical-chain`

The time after the unit is active or started is printed after the "@" character.  
The time the unit takes to start is printed after the "+" character.

```
graphical.target @1.505s
├─multi-user.target @1.505s
│   └─NetworkManager.service @1.473s +31ms
│       └─firewalld.service @800ms +672ms
│           └─basic.target @798ms
│               └─sockets.target @798ms
│                   └─sshd.socket @798ms
│                       └─sysinit.target @797ms
│                           └─systemd-timesyncd.service @763ms +33ms
│                               └─systemd-tmpfiles-setup.service @746ms +13ms
│                                   └─local-fs.target @745ms
│                                       └─boot.mount @731ms +13ms
│                                           └─systemd-fsck@dev-sda1.service @681ms +49ms
│                                               └─dev-sda1.device @680ms
```

87 / 281

## Débugger **systemd** ?

### Documentation :

- <http://freedesktop.org/wiki/Software/systemd/Debugging/>
- [https://fedoraproject.org/wiki/How\\_to\\_debug\\_Systemd\\_problems](https://fedoraproject.org/wiki/How_to_debug_Systemd_problems)
- <https://wiki.archlinux.org/index.php/Systemd#Troubleshooting>
- <https://wiki.debian.org/systemd#Debugging>

88/281

## Design et fonctionnalités : Standardisation

- Standardisation : chaque distribution avait ses propres scripts d'init et ses propres fichiers de configurations.
- Tout est maintenant commun :
  - `/etc/hostname`  $\Rightarrow$  `hostnamectl`;
  - `/etc/locale.conf`, `/etc/vconsole.conf`  $\Rightarrow$  `localectl`;
  - `/etc/localtime`  $\Rightarrow$  `timedatectl`.
- Mais pas encore supporté par toutes les distributions (attention notamment sur Debian & Ubuntu).

89/281

## `/run`

- **systemd** a introduit le dossier `/run`;
- Utilise le système de fichier `tmpfs` pour stocker les données temporairement;
- Disponible très tôt au démarrage;
- Utilisable pour stocker les informations non permanentes liées aux services;
- Voir **file-hierarchy(7)**.

90/281

## `tmpfiles.d`

- `{etc,usr/lib}/tmpfiles.d/*.conf`
- Liste de dossiers ou fichiers à créer automatiquement au démarrage;
- Exemple : `$ cat /usr/lib/tmpfiles.d/tuned.conf`

```
1 d /run/tuned 0755 root root -
```
- Voir **tmpfiles.d(5)**.

91/281

- `{etc,usr/lib}/sysusers.d/*.conf`
- Liste d'utilisateur système à créer automatiquement au démarrage si ils n'existent pas déjà ;
- Exemple : `$ cat /usr/lib/sysusers.d/mariadb.conf`  

```
1 u mysql - "MariaDB" /var/lib/mysql
```
- Voir **sysusers.d(5)**.

### 3 systemd

- Historique
- Fonctionnement
- journald
  - `logind`, `networkd`, `timesyncd`, `resolved`, etc.
  - Sécurité

- Similaire à `syslog(-ng)/rsyslog` : Récupère les messages envoyés par les processus sur la socket `/dev/log` et les stocke proprement ;
- Démon lancé très tôt au boot et fortement intégré avec `systemd` ;
- Cette intégration permet d'obtenir beaucoup plus d'informations que ce qui normalement possible avec `(r)syslog(-ng)` ;
- Utilise un format de stockage personnalisé et compressé [16] ;

- `journalctl -e` : affiche les logs les plus récents du boot courant ;
- `journalctl -b -1` : affiche les logs du boot précédent ;
- `journalctl -b -0 _PID=333` : limite l'affichage aux logs émis par le PID 333 pour ce boot ;
- `journalctl -k` : équivalent de `dmesg` ;
- `journalctl -u nginx(.service)` : affiche tous les logs liés au service `nginx` ;
- `journalctl -f -n 50` : affiche les 50 derniers logs puis affiche les logs au fur et à mesure ;
- `journalctl /usr/bin/dbus-daemon` : tous les logs émis par le binaire `/usr/bin/dbus-daemon`.



### 3 systemd

- Historique
- Fonctionnement
- journald
- logind, networkd, timesyncd, resolved, etc.
- Sécurité

- logind : gère les sessions utilisateurs (remplace ConsoleKit);
- networkd : gestion minimale « statique » du réseau pour ne plus dépendre de scripts à ce niveau;
- timesyncd : client NTP minimal (client uniquement);
- resolved : résolveur DNS;
- etc.

### 3 systemd

- Historique
- Fonctionnement
- journald
- logind, networkd, timesyncd, resolved, etc.
- Sécurité

- Support de toutes les options de gestion de processus proposées par le noyau :
  - UID, GID;
  - limites & Co.
  - cgroups v1 & v2;
  - namespaces;
  - capabilities;
  - seccomp;
  - etc.
- Exemples détaillés avec mise en place dans la présentation : Durcissement système à l'aide de systemd (SSTIC 2017) [17].

## Sécurité de systemd ?

- Quelques vulnérabilités majeures : CVE-2018-15686, CVE-2018-15688, CVE-2016-7795, CVE-2016-10156, CVE-2013-4392, CVE-2013-4327, CVE-2012-1174, CVE-2012-0871 ;
- Bugs médiatiques : CVE-2017-1000082, CVE-2017-9445, CVE-2017-15908 ;
- Écrit en C mais pratiquement toutes les erreurs sont logiques (non liées au langage) ;
- Surface d'attaque très limitée ;
- Démons systèmes correctement confinés (voir vulnérabilités resolved).

100/281

## Plan

- 1 Infrastructures
- 2 Conteneurs
- 3 systemd
- 4 Gestionnaires de conteneurs
  - Implémentations
- 5 Standards ?
- 6 Autres usages
- 5 SELinux
- 6 Conteneurs et sécurité
- 7 Virtualisation et sécurité
- 8 Sécurité des infrastructures

101/281

## Plan

- 4 Gestionnaires de conteneurs
  - Implémentations
  - Standards ?
  - Autres usages

102/281

## Création d'un conteneur sous Linux

Éléments nécessaires :

- gestion des ressources et de l'accès à certains périphériques avec les cgroups ;
- isolation (modulable) avec les namespaces ;
- restrictions de permissions avec les capabilities, les filtres seccomp et les Linux Security Modules (SELinux / AppArmor) ;
- image du conteneur (arborescence propre au conteneur : contenu du /) ;
- accès au réseau.

103/281

## Container runtime & Container engine

### Container runtime

Programme responsable de la création d'un conteneur et de la configuration de l'environnement d'exécution :

- mise en place des cgroups & limites UNIX ;
- mise en place des namespaces ;
- configuration des capabilities, filtres seccomp, LSM.

### Container engine

Programme responsable de la gestion du cycle de vie d'un conteneur :

- gestion des images de conteneurs (création, récupération à distance, envoi, instanciation, etc.) ;
- gestion des points de montage, volumes persistants, etc.
- mise en place des interfaces réseau (virtuelles) ;
- suivi des conteneurs en cours d'exécution.

Utilise un container runtime pour lancer les conteneurs.

**Attention :** Nomenclature et limite floue, souvent mélangé !

104/281

## Implémentations sous Linux

- LXC/LXD (2008) : LinuX Containers initialement développé par IBM. Maintenu par Ubuntu (Canonical) ;
- systemd-nspawn (2010) : Développé comme outils de test et remplaçant de chroot. Base de rkt. Maintenu ;
- Docker/Moby (2013) : Rends populaire les concepts de partage, copie, redistribution d'image de conteneurs. Maintenu par Docker Inc ;
- rkt (2014) : Créé par CoreOS pour corriger certains défauts et limitations de Docker. Futur incertain ;
- runc et containerd (2015) : Séparation de Docker en plusieurs projets indépendants ;
- Clear Containers (2015) : Créé par Intel, combine les conteneurs et la virtualisation matérielle avec KVM. Désormais regroupé avec runV dans le projet Kata Containers ;

105/281

## Implémentation sous Linux

- Minijail (2016) : Créé par Google pour ChromeOS & Android.
- railcar (2017) : Ré-implémentation de runc en Rust ;
- CRI-O (2017) : Ré-implémentation d'une partie de Docker pour usage avec Kubernetes uniquement. Maintenu par Red Hat ;
- Podman (2018), Buildah & Skopeo : Ré-implémentation des fonctionnalités de Docker sans démon. Maintenu par Red Hat ;
- gVisor (2018) : Isolation avec un noyau écrit en Go. Maintenu par Google ;
- Nabla containers (2018) : Isolation à l'aide d'un unikernel. Maintenu par IBM ;

106/281

## LXC/LXD

- Objectif initial : faire tourner un système complet dans un conteneur ;
- Image : simple tar.gz d'un /, obtenu avec debootstrap, dnf -install-root=... ou équivalent ;
- Création de conteneurs à partir de templates ;
- Avantages :
  - Terrain connu, fonctionnement prévisible ;
  - Aussi flexible qu'un système classique.
- Désavantages :
  - Duplication des services (journalisation, monitoring...) ;
  - Consommation en ressources (mémoire et disque) ;
  - Plus de gestion nécessaire (plus proche de la virtualisation « classique ») : mises à jour, sauvegarde...
- LXD : gestionnaire de conteneur LXC.

107/281

## Docker/Moby

- Objectif : faire tourner une seule application par conteneur ;
- Dockerfile : sorte de script pour créer des conteneurs ;
- DockerHub/Registry : partage d'images de conteneurs ;
- Avantages :
  - Simplicité (récupération des images, utilisation) ;
  - Consommation en ressources (mémoire et disque) réduite ;
  - Gestion à l'échelle d'une application et non plus d'un système.
- Désavantages :
  - Chaque conteneur doit être maintenu à jour indépendamment ;
  - Les applications classiques ne sont pas prévues pour fonctionner en tant que PID 1 dans un conteneur : processus zombie, etc.
  - Aucun service support n'est disponible dans le conteneur.

108/281

## Podman, Buildah, Skopeo

- Ré-écriture de Docker sans utiliser de démon central ;
- Buildah : Construction d'image de conteneur à partir d'un Dockerfile ou à l'aide de commande arbitraires ;
- Skopeo : Récupération d'images distante stockée dans un registre et envoi d'images locale ;
- Avantages :
  - Plus de problème liés au démon Docker : conteneur dans un conteneurs, redémarrage, etc.
  - Bonne intégration avec systemd ;
  - Implémentation de fonctionnalités très demandées mais jamais ajoutées à Docker ;
  - Séparation des tâches ⇒ meilleure sécurité.
- Désavantages :
  - N'implémente pas encore toutes les fonctionnalités de Docker ;
  - Moins « populaire » que LXC ou Docker.

109/281

## Plan

- 4 Gestionnaires de conteneurs
  - Implémentations
  - Standards ?
  - Autres usages

110/281

## Standards ?

- La prolifération des outils de gestion de conteneurs a poussé à la standardisation [92] ;
- Création de l'Open Container Initiative et de la Cloud Native Computing Foundation ;
- De facto standard : Docker Format v2 ;
- OCI Image Format Specification ;
- OCI Runtime Specification.

111/281

## Plan

### 4 Gestionnaires de conteneurs

- Implémentations
- Standards ?
- Autres usages

112/281

## Conteneurs et applications graphiques

- Sandboxing d'applications graphiques :
  - Firejail ;
  - Flatpak (& Flathub) ;
  - Snapcraft (& Snaps).
- Attention : Wayland nécessaire. Le serveur d'affichage X.org ne peut pas être configuré de façon sécurisé.
- A l'échelle d'un système d'exploitation :
  - Fedora Silverblue ;
  - Endless OS ;
  - Subgraph OS.

113/281

## Plan

### 1 Infrastructures

### 2 Conteneurs

### 3 systemd

### 4 Gestionnaires de conteneurs

### 5 SELinux

### ■ Concepts fondateurs

### ■ Modèles de contrôle d'accès

### ■ SELinux

### ■ SELinux en pratique

### 6 Conteneurs et sécurité

### 7 Virtualisation et sécurité

### 8 Sécurité des infrastructures

114/281

## Plan

### 5 SELinux

### ■ Concepts fondateurs

### ■ Modèles de contrôle d'accès

### ■ SELinux

### ■ SELinux en pratique

115/281

## La triade DIC (CIA triad)

### ■ Les fondamentaux :

- Confidentialité : qui a accès à une information ?
- Intégrité : qui peut écrire, modifier ou créer de l'information ?
- Disponibilité : est-ce qu'une ressource est disponible ?

### ■ Les extensions :

- Authenticité : comment établir une identité ?
- Auditabilité : comment tracer les opérations effectuées ?
- Non répudiation : est-il possible de nier avoir fait une action ?

116/281

## Le principe du moindre privilège

« Tous les programmes et utilisateurs d'un système doivent fonctionner en utilisant le minimum de privilèges nécessaires pour mener à bien leurs fonctions. »

— Jerome Saltzer, *Communications of the ACM*

Pour mettre en place un tel modèle d'opération, il faut :

- séparer les privilèges ;
- minimiser les privilèges ;
- confiner ou cloisonner les programmes privilégiés.

117/281

## Plan

### 5 SELinux

- Concepts fondateurs
- Modèles de contrôle d'accès
- SELinux
- SELinux en pratique

118/281

## DAC : Contrôle d'Accès Discretionnaire

- Le DAC est le modèle utilisé par défaut sous Linux ;
- Chaque utilisateur a le contrôle des fichiers et programmes lui appartenant :
  - création / modification / suppression
- Les programmes lancés disposent des mêmes droits que l'utilisateur
- Les utilisateurs peuvent donner des droits sur leurs ressources à leur *discretion* :
  - chmod / chown
- Le niveau de sécurité dépend de la bonne séparation entre utilisateurs et du niveau de sécurité de chaque application.

119/281

## Les limites du DAC

- Il est possible d'utiliser les *capabilities* et différents utilisateurs pour séparer les privilèges accordés aux applications ;
- Mais cette séparation n'est pas toujours assez fine :
  - de nombreuses *capabilities* sont équivalente à `root` (cf. section sur les *capabilities*) ;
- Il n'est pas possible de séparer des programmes s'exécutant sous un même identifiant UNIX.

Si programme s'exécutant sous l'identité `root` est compromis, l'attaquant obtient les droits `root` et l'ensemble du système est compromis.

120/281

## Contrôle d'accès obligatoire ou Mandatory Access Control (MAC)

- Les droits d'accès aux objets du système ne sont plus à la discrétion des utilisateurs ;
- L'administrateur du système définit une politique de sécurité qui décrit quel ressource est accessible à quel utilisateur.

121/281

## Contrôle d'accès obligatoire sous Linux

- Intervient après le contrôle d'accès discrétionnaire standard ;
- Implémenté sous forme de *Linux Security Module (LSM)* ;
- Implémentations disponibles sous Linux :
  - SELinux (utilisé par défaut sur Fedora, CentOS, RHEL) ;
  - AppArmor (utilisé par défaut sur Ubuntu, Debian) ;
  - SMACK (utilisé par défaut sur Tizen) ;
  - Tomoyo ;
  - grsecurity RBAC.

122/281

## Plan

- 5 SELinux
  - Concepts fondateurs
  - Modèles de contrôle d'accès
  - SELinux
  - SELinux en pratique

123/281

## SELinux ?

- Security-Enhanced Linux ;
- Implémentation d'un contrôle d'accès obligatoire ;
- Développé initialement par la NSA ;
- Premier MAC intégré dans le noyau Linux (2.6+) ;
- Implémentation du contrôle d'accès dans le noyau ;
- Outils en espace utilisateur pour manipuler la politique.

124/281

## SELinux & Android

- Utilisé par Android depuis la version 4.4 (KitKat), ~96% des *devices* ;
- Protection complète depuis la version 5.0 (Lollipop), ~89% des *devices* ;
- Impact significatif sur le durcissement du système [81,80].

125/281

## Concepts

- Basé sur l'architecture *Flask* :
  - Moniteur de référence dans le noyau ;
  - Stocke la politique de contrôle d'accès ;
  - Prends les décisions de contrôle d'accès au niveau des appels systèmes (contrôle à *grain fin*).

126/281

## Politique de sécurité

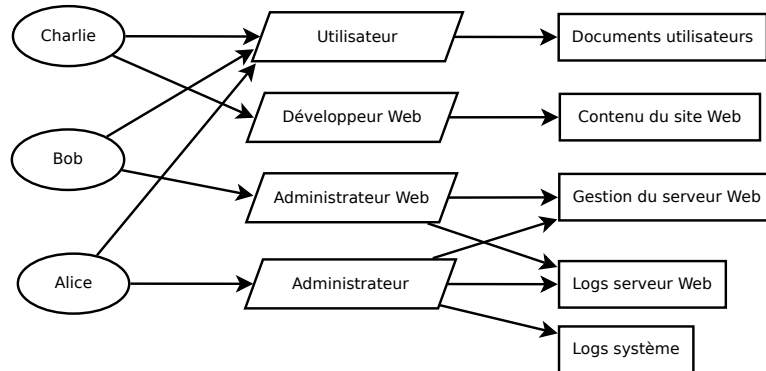
- Description exhaustive de l'ensemble des opérations autorisées sur un système ;
- Pour cela, on distingue :
  - Les sujets : entités actives du système (processus) ;
  - Les objets : entités passives du système (fichier, socket, etc.).
- La politique décrit quelles actions les sujets peuvent faire sur les objets ou sujets du système.

127/281



## Role Based Access Control (RBAC)

- Sous concept pour mettre en place le MAC ;
- Objectif : factoriser les opérations réalisables sur des objets en un rôle ;
- Association des utilisateurs à un ou plusieurs rôles.



128/281

## Politique

- Déclare les identiés, rôles et types ;
- Définition des règles d'autorisation d'accès ;
- Définition des règles de transitions de types ;
- Définition des contraintes, etc.
- Initialement écrit dans le Policy Language ;
- Vérifications à la compilation des contraintes ;
- Utilisation de macros/fonctions ;
- Remplaçant : SELinux Common Intermediate Language (CIL) [83] ;
- Même concepts, syntax différente.

129/281

## Type Enforcement (TE)

- SELinux associe à chaque ressource du système un contexte de sécurité (Security Context ou SC) ;
- Lorsqu'ils sont associés à des objets du système on les appelle aussi *label* ;
- Les décisions de contrôle d'accès prises par le noyau prennent en compte ces SC ;
- Dans le cas de SELinux, ce contrôle est appelé *Type Enforcement*.

130/281

## Intéraction

Chaque règle SELinux autorisant une interaction est constituée :

- d'un contexte de sécurité source (processus)
- qui effectue une opération :
  - class : file dir, process, socket (type de l'opération) ;
  - permission : read, write, unlink, accept (action) ;
- sur un contexte de sécurité destination (fichier, socket, processus, etc.).

131/281

## Contexte de sécurité

- Ensemble d'attributs sous forme de texte :
  - `system_u:system_r:httpd_t` : contexte du processus apache ;
  - `user_u:object_r:user_home_t` : contexte de fichier d'un répertoire utilisateur.
- Trois attributs principaux : `user:role:type`
  - User : identité SELinux ( $\neq$  DAC) propriétaire du contexte ;
  - Role : modèle RBAC
    - rôle courant de l'identité pour un sujet ;
    - rôle par défaut pour les objets (`object_r`).
  - Type : modèle TE
    - Domaine dans lequel s'exécute un sujet ;
    - Type associé à un objet.

132/281

## Contexte de sécurité

- Pour les processus :
  - attribué à la création ;
  - ne peut évoluer que lors d'un appel à `execve`.
- Pour les fichiers :
  - stocké dans les attributs étendus du fichier (`xattr`) ;
  - peut être modifié à volonté.

133/281

## Type

- Un type est un identifiant ;
- Pas de signification intrinsèque : il est contraint aux règles de la politique ;
- Regroupe les sujets et objets ayant les mêmes autorisations.
- On parle de :
  - type pour un objet ;
  - domaine pour un sujet.
- Exemples : `system_t`, `user_t`, `unconfined_t`

134/281

## Convention de nommage

- `applicationd_t` : domaine associé à un démon ;
  - `application_t` : domaine associé à un processus classique ;
  - `application_exec_t` : fichier binaire de l'application ;
  - `application_conf_t` : fichier de configuration ;
  - `application_log_t` : fichier de log ;
  - `application_file_t` : fichiers divers ;
  - `application_lib_t` : bibliothèque ;
  - `application_tmp_t` : fichier temporaire ;
  - `applicationd_unit_file_t` : fichier d'unit systemd.
- Un objet/processus ne possède qu'un seul type mais un type peut être associé à plusieurs objets.

135/281

## Types

Chaque type doit être déclaré avec utilisation :

```
1 # sujet
2 type init_t;
3 type sshd_t;
4
5 # objet
6 type sshd_exec_t
7 type admin_passwd_exec_t;
```

136/281

## Attributs

Attributs : regroupe plusieurs types pour factoriser la politique :

```
1 # déclaration d'un attribut
2 attribute file_type;
3 attribute application_domain_type;
4 attribute application_exec_type;
5
6 # association d'un type (déjà déclaré) à un attribut (déjà
  ↳ déclaré)
7 typeattribute shadow_t security_file_type;
8
9 # déclaration d'un type et association à un attribut (déjà
  ↳ déclaré)
10 type sshd_exec_t executable_file_type;
```

137/281

## Opération

- Opération = droit d'un contexte sujet sur un autre contexte;
- Permission autorisée sur une classe d'objet;
- Exemple :
  - file : { open read execute };
  - socket : { bind listen }.

138/281

## Règle de contrôle d'accès

```
allow <type_source> <type_cible>:<class> { <permissions> };
1 # Les sujets avec le type sysadm_t peuvent exécuter (...)
  ↳ les fichiers avec le type ssh_exec_t
2 allow sysadm_t ssh_exec_t:file {getattr read execute };
3
4 # Les sujets avec le type sysadm_t peuvent exécuter (...)
  ↳ les fichiers dont le type a pour attribut
  ↳ application_exec_type
5 allow sysadm_t application_exec_type:file {getattr read
  ↳ execute ioctl lock execute_no_trans };
```

139/281

## Factorisation des permissions

- Macros pour simplifier l'utilisation des permissions :  
policy/support/\*perms\_set.spt
- Exemples :
  - r\_file\_perms : permissions pour lire un fichier
    - open, getattr, read, lock, ioctl
  - r\_dir\_perms : permissions pour traverser un répertoire
    - open getattr read lock search ioctl
  - x\_file\_perms : permissions pour exécuter un fichier
    - open getattr execute

140/281

## Type & file transitions

- Par défaut, le contexte de sécurité d'un nouveau processus est celui de son père ;
  - Possibilité de changer de contexte avec une *transition* lors d'un appel à `execve` ;
  - Permet de confiner chaque service dans son domaine respectif.
- 
- Par défaut, le contexte de sécurité d'un fichier est celui de son répertoire parent.
  - Possibilité d'ajouter des règles pour associer des contextes différents en fonction des répertoire ou noms de fichiers.

141/281

## Transition de type : sujet

```
1 # Lorsque le type user_t crée exécute le fichier ayant le
   ↳ type ssh_exec_t il transite automatiquement vers
   ↳ user_ssh_t
2 type_transition user_t ssh_exec_t:process user_ssh_t;
3
4 # Lorsque le type user_t crée exécute le fichier ayant le
   ↳ type passwd_exec_t il transite automatiquement vers
   ↳ passwd_t
5 type_transition user_t passwd_exec_t:process passwd_t;
6
7 # Règles supplémentaires correspondantes
8 allow user_t ssh_exec_t : file {read getattr execute}
9 allow user_ssh_t ssh_exec_t : file entrypoint ;
10 allow user_r user_ssh_t : process transition ;
```

142/281

## Transition de type : objets

```
1 # Lorsque le processus labellé par user_t accède à un
   ↳ fichier (...) de type tmp_t, le type de cet objet
   ↳ transite automatiquement vers user_tmp_t
2 type_transition user_t tmp_t:{ dir file lnk_file sock_file
   ↳ fifo_file } user_tmp_t;
3
4 # Lorsque le processus labellé par sshd_t accède à un
   ↳ fichier (...) de type tmp_t, le type de cet objet
   ↳ transite automatiquement vers sshd_tmp_t
5 type_transition sshd_t tmp_t:{ dir file sock_file }
   ↳ sshd_tmp_t;
6
7 # Règles supplémentaires correspondantes
8 allow user_t tmp_t: dir { open getattr create write };
9 allow user_t user_tmp_t:file rw_file_perms;
10
11 # Lorsque qu'un processus user_t crée un dossier nommé .ssh
   ↳ dans un dossier user_home_dir_t, il est labellé
   ↳ ssh_home_t
12 type_transition user_t user_home_dir_t : dir ssh_home_t
   ↳ .ssh;
```

143/281

## Politique Multi-Niveau (MLS & MCS)

Deux attributs supplémentaires ajoutés aux contextes de sécurité :

`unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023`

### ■ MLS : niveaux de sécurité

- Ajout de contrainte entre différents niveau de sécurité;
- Souvent utilisé pour implémenter des modèles de confidentialité.
- N'est pas utilisé dans la politique *targeted*.

### ■ MCS : catégories

- Isole plusieurs instances d'un même type;
- Le ' ' définit un ensemble (ex : `c0.c16`);
- La ', ' définit une liste (ex : `c21,c36,c45`);
- Les deux séparateurs peuvent être combinés (ex : `c0.c16,c21,c36,c45`).

144/281

## Politique Multi-Niveau (MLS & MCS)

- Contrôle entre les niveaux et les catégories est réalisé via des contraintes SELinux;
- Décrites dans la politique : `policy/mls`, `policy/mcs`;
- Tous les types ne sont pas nécessairement contraints;
- Utilisé principalement pour confiner les machines virtuelles et les conteneurs (cf. chapitre sécurité & conteneurs, sécurité & KVM).

145/281

## Plan

### 5 SELinux

- Concepts fondateurs
- Modèles de contrôle d'accès
- SELinux
- SELinux en pratique

146/281

## Refpolicy

- Strict policy;
- Tous les utilisateurs sont confinés;
- La politique doit décrire l'intégralité des interactions autorisées;
- Disponible principalement sous Gentoo Hardened (Debian & Arch Linux);
- Restrictif et complexe.

147/281

- Targeted policy ;
- Les démons système sont confinés ;
- Les utilisateurs interactifs ne sont pas confinés par défaut (unconfined) ;
- Possibilité de confiner certains utilisateurs.

- Configuration de SELinux et politique actuelle stockées dans `/etc/selinux/` ;
- `sestatus -v` : état général ;
- État actuel de SELinux dans le noyau : `/sys/fs/selinux/` ;
- État actuel de la politique stocké dans `/var/lib/selinux/` ;
- `seinfo -v` : informations sur la politique.

- SELinux utilise le sous système d'audit du noyau ;
- Démon `auditd` en espace utilisateur ;
- Logs dans `/var/log/audit/audit.log` ou dans le journal.
- Loggue par défaut tous les accès refusés ;
- Outils `audit2allow` et `audit2why` pour générer des règles de politique à partir de ces logs.

```
1 type=AVC msg=audit(1515257682.718:409): avc: denied {  
  ↳ getattr } for pid=4320 comm="httpd"  
  ↳ path="/home/fedora/public_html/test" dev="sda1"  
  ↳ ino=262285 scontext=system_u:system_r:httpd_t:s0  
  ↳ tcontext=unconfined_u:object_r:httpd_user_content_t:s0  
  ↳ tclass=file permissive=0  
2 type=AVC msg=audit(1515257956.228:527): avc: denied {  
  ↳ setuid } for pid=4646 comm="sudo" capability=7  
  ↳ scontext=user_u:user_r:user_t:s0  
  ↳ tcontext=user_u:user_r:user_t:s0 tclass=capability  
  ↳ permissive=0
```

## Règle de contrôle d'accès : audit

```
1 # Audite l'accès en lecture au type shadow_t par sysadm_t
2 auditallow sysadm_t file:read shadow_t;
3
4 # Les types non privilégiés ne peuvent pas lire shadow_t
5 # Utilisation de ~ pour la négation
6 neverallow ~can_read_shadow_passwords shadow_t:file read;
7
8 # Ne pas auditer certains accès aux répertoires pour le
9   ↳ type staff_t
10 # Utilisation de - pour retirer le type security_file_type
11   ↳ de file_type
12 dontaudit staff_t { file_type -security_file_type }: dir {
13   ↳ getattr search read lock};
```

152/281

## Politique modulaire

- Politique source stockée sous forme d'une collection de fichiers texte;
  - Politique séparée entre une base et de nombreux modules;
  - « Compilation » de la base et des modules pour former la politique finale, sous forme binaire.
  - Seule la forme finale est chargée par le noyau.
- 
- On écrit rarement une politique de zéro;
  - On ajoute ou modifie un module existant.

153/281

## Module SELinux

- Composé de trois fichiers :
  - .te : ensemble des règles SELinux à appliquer;
  - .fc : list des associations entre chemin de fichiers et contexte (*fc* : *file contexts*);
  - .if : interface, ou ensemble de « fonctions » proposé pour utilisation dans d'autres modules.
- Création d'un module à l'aide de `/usr/share/selinux/devel/Makefile`;
- Installation avec `semodule -install -priority=500`.

154/281

## Exemple de module SELinux

```
toto.te :
1 policy_module(toto, 0.0.1)
2
3 require {
4     type httpd_t;
5     type httpd_sys_script_t;
6     type user_tmp_t;
7 }
8
9 type toto_data_t;
10 files_type(toto_data_t);
11
12 allow httpd_t toto_data_t:file read_file_perms;
13 allow httpd_t toto_data_t:dir search_dir_perms;
14
15 allow httpd_sys_script_t user_tmp_t:file read_file_perms;
```

155/281

## Exemple de module SELinux

toto.fc :

```
1 /var/www/html/toto          -d
   ↪ gen_context(system_u:object_r:toto_data_t,s0)
2 /var/www/html/toto(/.*)?    --
   ↪ gen_context(system_u:object_r:toto_data_t,s0)
```

156/281

## Ecrire un module SELinux

- ① `sepolgen` : génère un canevas à partir de template pour les modules couramment ajoutés.
- ② « Compilation » du module ;
- ③ Chargement du module ;
- ④ Tests ;
- ⑤ Lecture des logs ;
- ⑥ Modification du module ;
- ⑦ Goto 2.

Fin : Désactivation du mode permissif.

157/281

## Contextes de sécurité

### ■ Pour les processus :

- `id -Z;`
- `ps -Z.`

### ■ Pour les fichiers :

- `ls -Z;`
- `mkdir -Z;`
- `cp -Z;`
- `mv -Z;`
- `find -context.`

158/281

## Gestion des contextes

- `restorecon` : restaure le contexte associé à un fichier par la politique :
  - `-R` : récursif ;
  - `-v` : verbeux ;
  - `-F` : restaure aussi le rôle et l'utilisateur SELinux.
- `chcon` : modification arbitraire du contexte d'un fichier.
  - `-u, -r, -t` : définit l'utilisateur, rôle, type ;
  - `-R` : récursif ;
  - `-reference=file` : utilise le contexte d'un autre fichier comme référence.
- `fixfiles onboot` : restaure les contextes associés à tous les fichiers du système au prochain boot.

159/281



## Points de montage

- Options à passer à mount :
- Force le contexte pour tous les fichiers : `-o context=user:role:type`
- Contexte par défaut pour les fichiers sans contexte : `-o defcontext=user:role:type`
- Contexte pour le système de fichier : `-o fscontext=user:role:type`

160/281

## Manipulation de la politique

- `semanage` : gestion de la politique SELinux
  - `login` : association entre les utilisateurs Linux et SELinux ;
  - `user` : utilisateurs SELinux
  - `port` : association contextes et ports réseau
  - `module` : modules de la politique
  - `fcontext` : associations contextes et fichiers
  - `boolean` : gestion des booléens ;
  - `dontaudit` : active/désactive les règles *dontaudit*.
  - etc.
- `semodule -DB` : désactiver les règles *dontaudit*.

161/281

## Booléens

- Une fois chargée, la politique est immuable ;
- Possibilité d'ajouter des conditions pour autoriser dynamiquement certaines opérations sans avoir à changer la politique ;
- Commandes `getsebool` & `setsebool` OU `semanage boolean`.

162/281

## Recherches dans la politique

- `sesearch` : recherche de règles dans la politique.
- Liste tous les fichiers que le contexte `httpd_t` peut lire :
  - `sesearch -v -allow -s httpd_t -c file -p read`
- Liste tous les contextes pouvant écrire dans les fichiers `shadow_t` :
  - `sesearch -v -allow -t shadow_t -c file -p write`
- Liste toutes les règles associées au booléen `samba_enable_home_dirs` :
  - `sesearch -v -allow -b samba_enable_home_dirs`

163/281

## Inspection de la politique

- `sepolicy` :
  - `booleans` : informations sur les booléens ;
  - `communicate` : est-ce que deux domaines peuvent interagir entre eux ?
  - `generate` : générer des modules à partir de template ;
  - `gui` : lance les outils graphiques ;
  - `interface` ;
  - `manpage` : générer les pages de man de SELinux ;
  - `network` : relation entre ports et types SELinux ;
  - `transition` : est-ce qu'un domaine peut transiter vers un autre ?

164/281

## Désactiver SELinux ?

- Pour désactiver complètement SELinux, il faut redémarrer le système ;
- `selinux=0` sur la ligne de commande du noyau ou configuration dans `/etc/selinux/config`.
- Plus de MAC, fonctionnement classique DAC.
- Avant de réactiver SELinux, il faut relabéliser intégralement le système de fichiers (`fixfiles onboot`) et redémarrer.

165/281

## Désactiver SELinux ?

- `setenforce 0` ;
- Mode *Permissive*, qui n'applique plus les contrôles mais loggue uniquement les accès refusés ;
- Le mode permissif ne doit pas être activé sur un système entier en production ;
- Si un démon ne fonctionne pas correctement avec SELinux, il faut passer uniquement ce démon en permissif :
  - `semanage permissive -l`
  - `semanage permissive -a myapp_t`
  - `semanage permissive -d myapp_t`

166/281

## Plan

- |                               |   |
|-------------------------------|---|
| 1 Infrastructures             | 6 Conteneurs et sécurité <ul style="list-style-type: none"><li>■ Sécurité de l'hôte</li><li>■ Sécurité de l'image</li><li>■ Sécurité de l'application</li></ul> |
| 2 Conteneurs                  |   |
| 3 systemd                     |   |
| 4 Gestionnaires de conteneurs | 7 Virtualisation et sécurité  |
| 5 SELinux                     | 8 Sécurité des infrastructures  |

167/281

## Points d'attention

- Sécurité de l'hôte hébergeant les conteneurs :
  - Matériel ;
  - Noyau ;
  - OS ;
  - Gestionnaire de conteneurs ;
  - Isolation entre les conteneurs.
- Sécurité de l'image du conteneur :
  - Distribution et intégrité ;
  - Distribution des secrets.
- Sécurité de l'application dans le conteneur :
  - Langage de programmation & Framework ;
  - Système de base ;
  - Mise à jour.

168/281

## Plan

- 6 Conteneurs et sécurité
  - Sécurité de l'hôte
  - Sécurité de l'image
  - Sécurité de l'application

169/281

## Matériel

### Matériel sécurisé ?

- Intel Management Engine ;
- AMD Platform Security Process ;
- Attacking SMM Memory via Intel CPU Cache Poisoning [62].

170/281

## Matériel

### Attaques par canaux cachés :

- Pas d'isolation physique : matériel partagé ;
- Attaques utilisant des canaux cachés (ou détournées) [45], [46] :
  - temps & cache : Meltdown & Spectre [60] ;
  - son (GnuPG [47]) ;
  - bruit électromagnétique
  - ...
- Concerne particulièrement les opérations cryptographiques.

171/281

Se reposer sur du matériel de confiance :

- TPM : What Trusted Computing Means to Users of CoreOS and Beyond [91].

- Point commun entre tous les conteneurs ;
- Faille exploitable à distance : souvent réseau, très rare ;
- Faille locale : appels systèmes, surface d'attaque large :
  - Gestion de la mémoire (Dirt COW [20]) ;
  - Systèmes de fichiers (overlayfs : [10], partage de /proc et /sys [11]) ;
  - Drivers : ioctl (cf. Android), ...
  - Réseau : netfilter...
- Retirer des appels systèmes (dur) ;
- Limiter les appels systèmes disponibles : seccomp.

- L'isolation fournie par les namespaces et les cgroups a encore des limites ;
- Certaines parties du noyau n'ont pas encore été adaptées aux namespaces ;
- Beaucoup de vulnérabilités trouvées ces dernières années : CVE-2013-1858, [4], [5],[6], [7], [8], [9] ;
- Noyau : cible privilégiée car commun à tous les namespaces ;
- Présentations sur l'exploitation du noyau : [18] et [19].
- Cas non prévus lorsque l'on combine plusieurs namespaces ;
- Failles noyau donc très souvent critiques (« local root ») ;
- User namespace particulièrement touchés.

- Initialement : PaX & grsecurity ;
- Aucun patch public et maintenu disponible.
- Travail en cours par le *Kernel Self Protection Project* pour intégrer ces améliorations upstream.

## Système d'exploitation

- Mises à jour;
- Durcissement standard :
  - compilation durcie (stack protector, RELRO, etc.);
  - recommandations génériques [70,73].

176/281

## Outils de gestion de conteneurs

- Tous les outils d'infrastructure pour gérer les conteneurs augmentent la surface d'attaque.

177/281

## Vulnérabilités spécifiques à LXC

- Vulnérabilité à la création d'un conteneur [30];
- Vulnérabilité dans les templates de création de conteneurs [31].

178/281

## Vulnérabilités spécifiques à Docker

### Sécurité ?

« Docker is about running random code downloaded from the Internet and running it as root. »

179/281

## Vulnérabilités spécifiques à Docker

- Démon docker tournant en root ;
- Récupération d'images depuis le réseau [35];
- Opération sur des images qui ne sont pas de confiance [32];
- Beaucoup d'images proposés sur le Registry ne sont pas à jour [36];
- Signatures des images pas avant la version 1.8 [33], [34];
- Plusieurs vulnérabilités : [37], [38], [39];
- Nouvelle architecture depuis la version 1.11.0.

180/281

## Spécificités Docker

- Accès au démon docker  $\Leftrightarrow$  root ;
- Restreindre impérativement l'accès au démon docker : [86];
- Plusieurs guides pour utiliser Docker correctement : [84], [85].
- Ne concerne pas Podman (pas de démon).

181/281

## Isolation : SELinux

- Objectif : contenir les conteneurs ;
- Utilise les catégories (MCS);
- Tous les conteneurs utilisent le même type (même accès);
- Associe à l'exécution un ensemble de catégories à chaque conteneur ;
- SELinux garantie qu'il ne peut pas y avoir d'interaction entre les conteneurs si leur ensemble de catégories sont distincts ;
- Supporté par Docker, rkt et Podman [78], [77], [79];
- Limite la portée d'une grande partie des vulnérabilités : ex. CVE-2016-9962 ;
- AppArmor supporté par Docker et Podman.

182/281

## Plan

- 6 Conteneurs et sécurité
  - Sécurité de l'hôte
  - Sécurité de l'image
  - Sécurité de l'application

183/281

## Sécurité de l'image

- Restriction des permissions :
  - Utiliser un utilisateur non privilégié ;
  - Limiter les *capabilities* ;
  - Appliquer un filtre *seccomp* strict.
- Image du conteneur en lecture seule (RO) ;
- Stockage persistant externe (par ex. volumes Docker) pour :
  - la configuration en lecture seule ;
  - les données en lecture / écriture.

184/281

## Sécurité de l'image

- Contrôle de l'intégrité du conteneur :
  - stockage ;
  - transport sur le réseau ;
  - vérification avant exécution.
- Distribution des secrets ?
  - Considérer une image de conteneur comme publique ;
  - Ne pas inclure de secrets dans une image de conteneur ;
  - Utilisation de variables d'environnement ;
  - Récupération dynamique des secrets : Vault [88].

185/281

## Plan

- 6 Conteneurs et sécurité
  - Sécurité de l'hôte
  - Sécurité de l'image
  - Sécurité de l'application

186/281

## Sécurité des applications : langages

- Une application vulnérable sera toujours vulnérable dans un conteneur ;
- Utiliser des langages de programmations et des Frameworks avec des propriétés intéressantes en terme de sécurité :
  - Rust, Go, Kotlin ;
  - Haskell, OCaml, Java, Elixir, Erlang, Python.

187/281

## Sécurité des applications : système de base

- Conteneurs sont souvent créés à partir d'un système de base ;
- La taille n'est pas toujours un facteur de décision (même si l'on cherche souvent à la réduire au maximum) ;
- Choisir un système de base avec des bonnes propriétés de sécurité et correctement supporté :
  - Attention aux images basées sur Alpine :
    - Compatibilité avec musl libc et LibreSSL ;
    - Gestionnaire de paquet spécifique (apk) [89].
  - Les distributions standard (Debian, Ubuntu, Fedora, Red Hat/CentOS) restent une valeur sûre.

188/281

## Sécurité des applications : mises à jour

- Conteneur  $\Leftrightarrow$  énorme binaire statique ;
- Absolument prévoir un cycle de mise à jour ;
- Intégration Continue (CI) ;
- Outils pour vérifier qu'une image ne contient pas de vulnérabilité déjà connue :
  - Clair [114] ;
  - Open Scap [115].

189/281

## Plan

- 1 Infrastructures
- 2 Conteneurs
- 3 systemd
- 4 Gestionnaires de conteneurs
- 5 SELinux
- 6 Conteneurs et sécurité
- 7 Virtualisation et sécurité
  - Virtualisation avec KVM/QEMU
  - Scénarios d'attaque et vulnérabilités
  - Virtualisation et sécurité
- 8 Sécurité des infrastructures

190/281

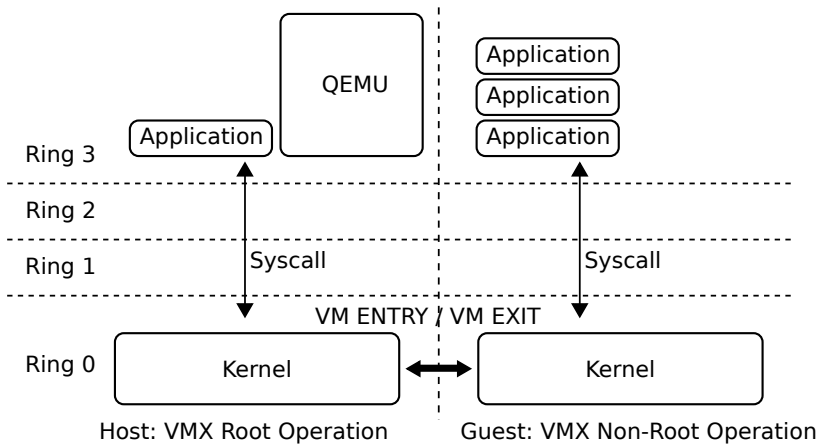
## Plan

- 7 Virtualisation et sécurité
  - Virtualisation avec KVM/QEMU
  - Scénarios d'attaque et vulnérabilités
  - Virtualisation et sécurité

191/281

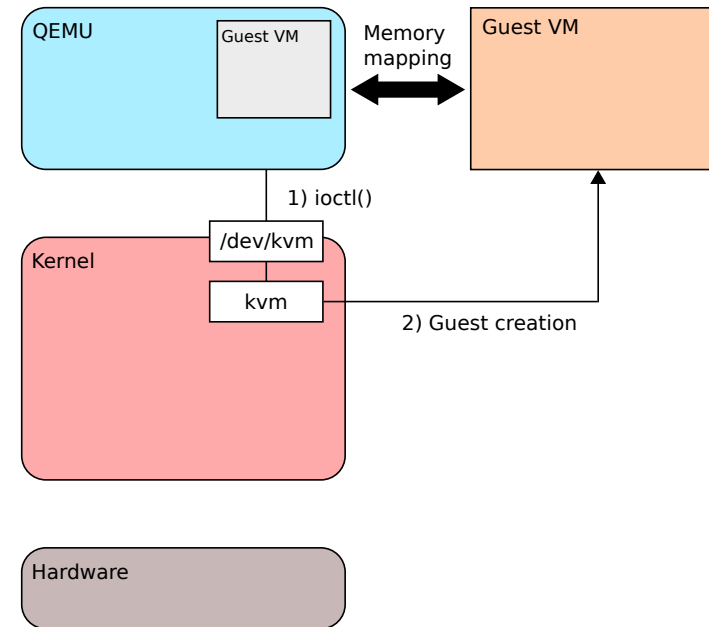


## Principe général : KVM & VT-x (Intel)



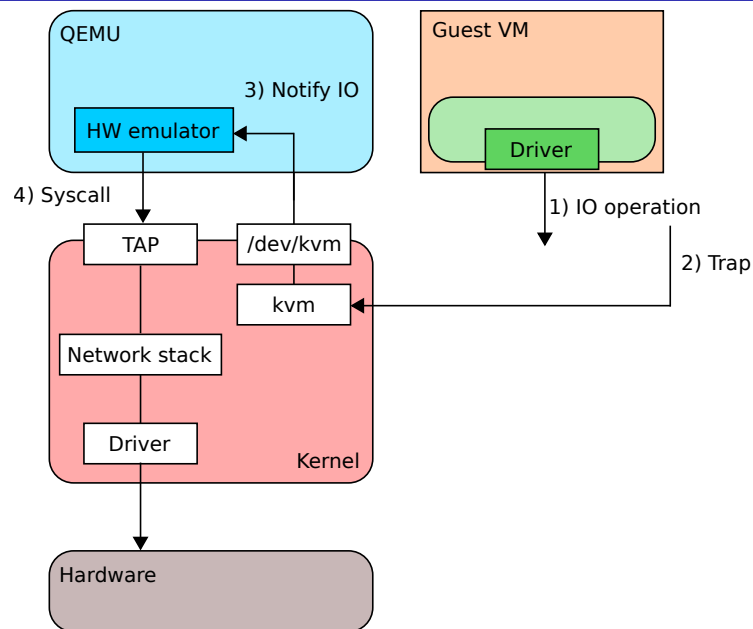
192/281

## Création d'une VM avec QEMU



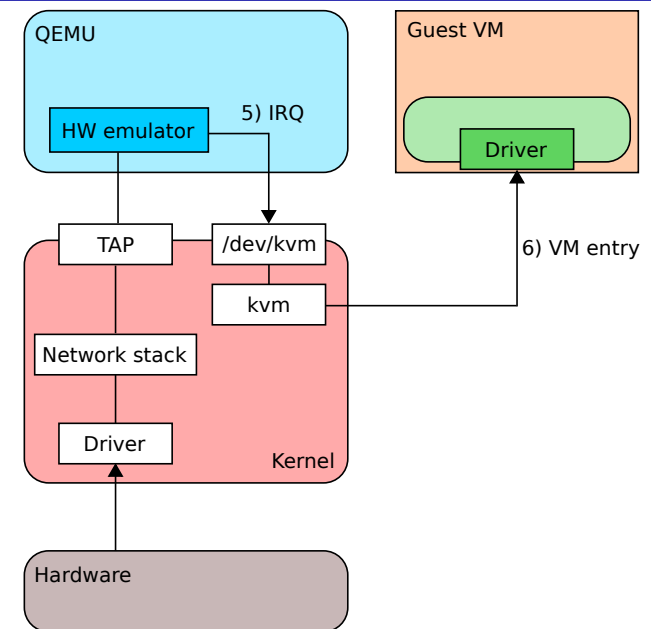
193/281

## Matériel émulé (QEMU) I



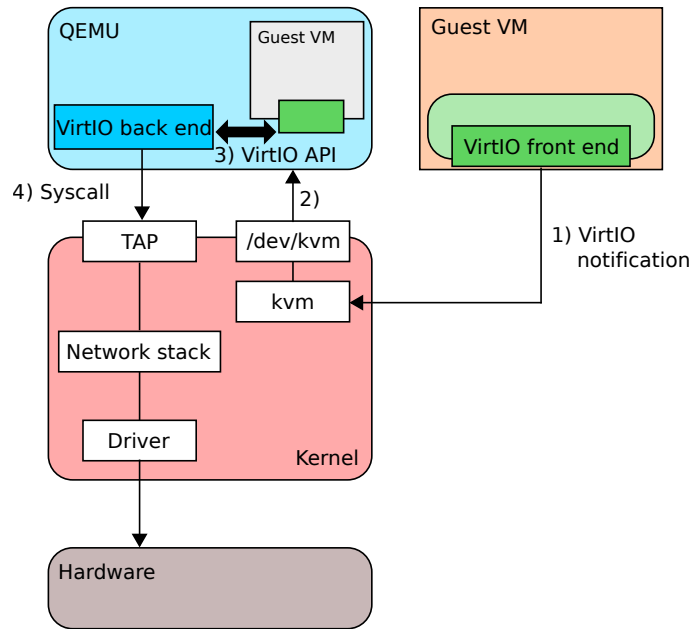
194/281

## Matériel émulé (QEMU) II



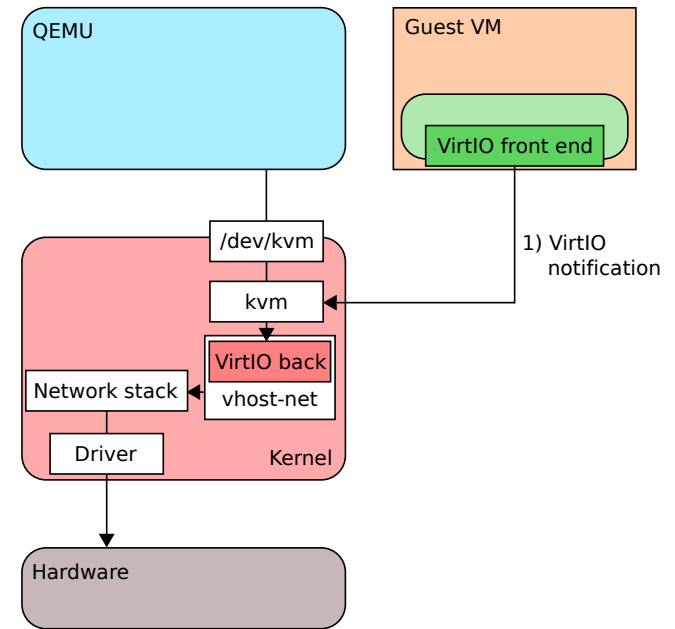
195/281

## Matériel virtuel : VirtIO (QEMU)



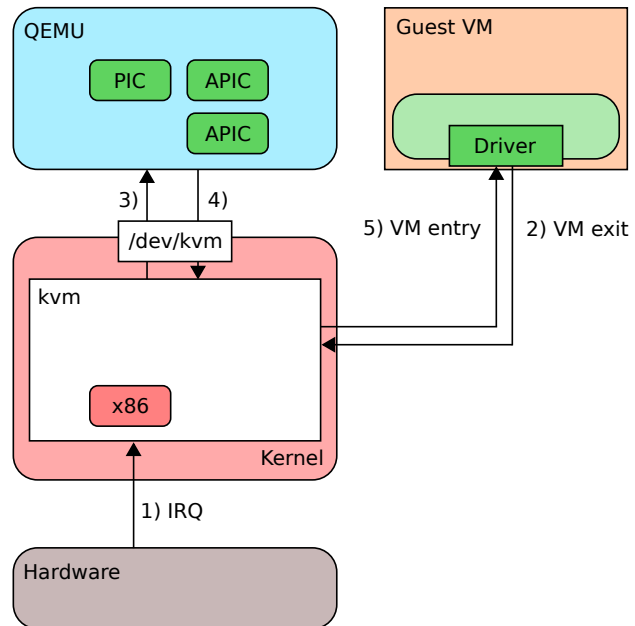
196/281

## Matériel virtuel : VirtIO (Kernel)



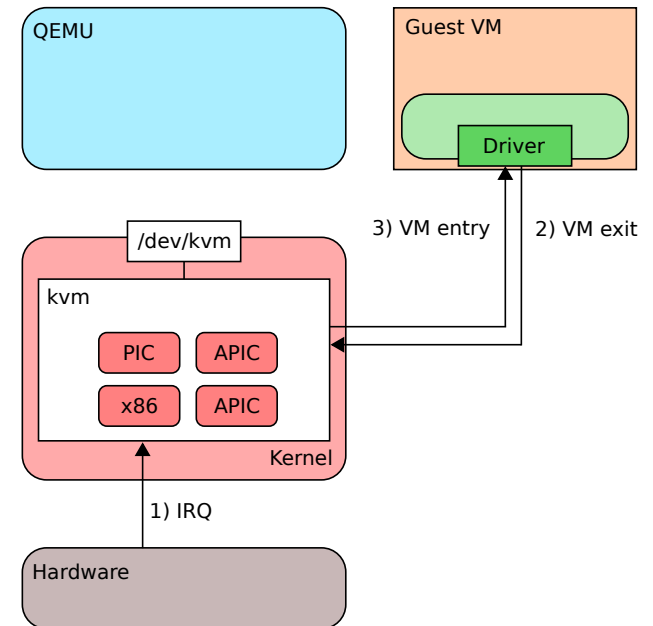
197/281

## Composants émulés (QEMU)



198/281

## Composants émulés (Kernel)



199/281

### 7 Virtualisation et sécurité

- Virtualisation avec KVM/QEMU
- Scénarios d'attaque et vulnérabilités
- Virtualisation et sécurité

La virtualisation n'est pas un composant de sécurité !

Plus d'isolation, mais :

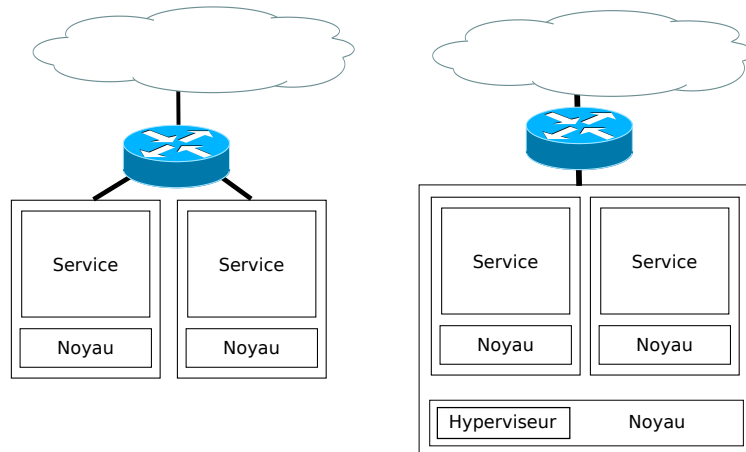
- Matériel plus compliqué ;
- Plus de versions de systèmes (noyau, espace utilisateur) ;
- Plus de réseaux à gérer ;
- Plus d'interfaces d'administration.

- La virtualisation ne dispense donc pas des autres solutions pour assurer la sécurité ;
- Au contraire, elle en ajoute à appliquer.

## Virtualisation == sécurité ?

Pour s'en convaincre, il faut comparer :

- Services sur des machines physiquement distinctes vs machines virtuelles ;

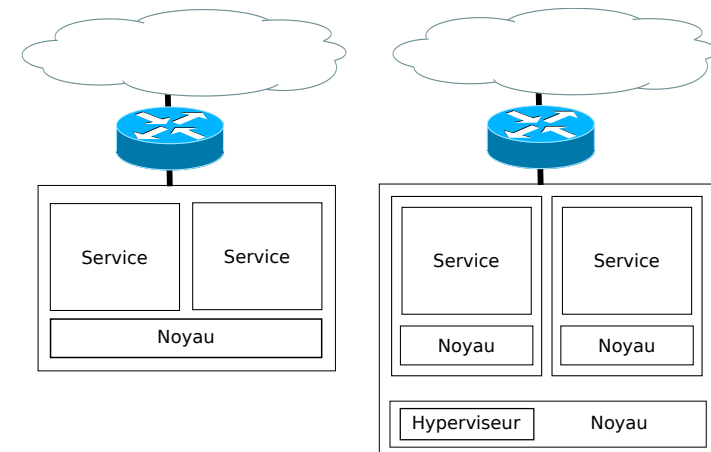


204/281

## Virtualisation == sécurité ?

Pour s'en convaincre, il faut comparer :

- Services sur une même machine vs services dans une ou plusieurs machines virtuelles.



205/281

## Scénario d'attaque : Machine virtuelle

- La machine virtuelle (le système, les services ou les données) sont visés ;
- Principalement des attaques à distance similaires aux attaques classiques pour les machines physiques ;
- Les machines virtuelles ne sont pas exemptées des contraintes de sécurité appliquées aux machines physique ;

206/281

## Scénario d'attaque : Autres VMs

- Corrompre/espionner les autres machines virtuelles sur le même hyperviseur ;

207/281

## Scénario d'attaque : Système hôte

- Sortir de la machine virtuelle pour attaquer le système hôte et les autres machines virtuelles ;
- Attaquer ou corrompre l'hyperviseur :
  - faille dans l'hyperviseur ou les drivers virtuels ;
  - faille dans le système hôte.

208/281

## Scénario d'attaque : Virtualisation à la volée

- Virtualiser à la volée un système non-virtualisé sans visibilité pour l'utilisateur ;
- Théoriquement possible (Blue Pill [48]) ;
- Invisibilité contestée en pratique : il est très difficile de masquer à un système qu'il est virtualisé.

209/281

## Xen

- XSA-109 [55], XSA-123 [56] : élévation de privilège ;
- XSA-148 [57], [58] :
  - Nécessite d'être `root` dans une machine virtuelle PV ;
  - Accès à la totalité de la mémoire du système.
- XSA-155 [59] :
  - Nécessite d'être `root` dans une machine virtuelle PV ou HVM ;
  - Élévation de privilège.

210/281

## QEMU

- CVE-2011-1751 (aka Virtunoid) [49] :
  - Bug dans la gestion du temps dans QEMU ;
  - Nécessite d'être `root` la machine virtuelle ;
  - Prise de contrôle de QEMU en Ring 3 mode VMX Root (« sortir » de la machine virtuelle).
- CVE-2015-3456 (aka VENOM) [50] :
  - Bug dans le pilote du contrôleur de disquettes émulé par QEMU ;
  - Impacte tous les hyperviseurs ;
  - Activé par défaut dans toutes les machines virtuelles sans possibilité de le désactiver à l'exécution ;
  - Nécessite d'être `root` la machine virtuelle ;
  - Prise de contrôle de QEMU en Ring 3 mode VMX Root (« sortir » de la machine virtuelle).

211/281

- CVE-2015-0239 : Émulation incomplète de l'instruction SYSENTER [63];
- CVE-2014-0049 : Erreur dans l'émulateur x86 [64].

- Faille driver VirtIO - CVE-2011-4127 [65] :
  - Nécessite d'être root la machine virtuelle ;
  - Donne l'accès total à un disque physique SCSI de l'hôte car le noyau ne vérifiait pas que les commandes SCSI correspondaient uniquement à des parties accessibles à la VM ;
  - Faille dans le noyau hôte.

- Il est très fortement déconseillé de monter directement le système de fichiers d'une machine virtuelle sur la machine hôte [67] :
  - C'est mount qui détermine quel module doit être chargé en fonction du système de fichiers : il peut donc être trompé ;
  - Une faille dans un driver de système de fichier ou dans la couche VFS du noyau Linux peut compromettre toute la machine :
- Il faut donc utiliser d'autres outils qui le font indirectement : libguestfs.

- 7 Virtualisation et sécurité
  - Virtualisation avec KVM/QEMU
  - Scénarios d'attaque et vulnérabilités
  - Virtualisation et sécurité

## Risques principaux

- Sécurité de l'hyperviseur :
  - Matériel / Noyau / OS ;
  - Gestionnaire de machines virtuelles (VMM) ;
  - Emulateur / virtualiseur ;
  - Isolation entre les machines virtuelles.
- Sécurité des machines virtuelles ;
  - Noyau / OS ;
  - Applications ;
  - Mise à jour.
- Sécurité des images de machines virtuelles :
  - Distribution et intégrité ;
  - Distribution des secrets.

216/281

## Matériel

- Matériel sécurisé ?
- Canaux cachés ;
- Matériel de confiance ;
- Cf. chapitre sur la sécurité des conteneurs.

217/281

## Matériel : virtualisation

Surface d'attaque supplémentaire :

- Fonctions de virtualisation dans les processeurs ;
- CPU et MMU : SVM/AMD-V (AMD), VT-x (Intel) ;
- IOMMU : AMD-Vi (AMD), VT-d (Intel).

218/281

## IOMMU & PCI passthrouh

- IOMMU intéressante dans tous les cas : contrôle d'accès entre périphériques PCI-Express ;
- PCI passthrough et SR-IOV à éviter [66] :
  - firmware du matériel accessible ?
  - interfaces cachées ?
  - quelle isolation entre les différents contextes par le matériel ? (exemple : cartes graphiques [44])
  - ...
- Utiliser VFIO.

219/281

## Noyau

- Cf. chapitre sur la sécurité des conteneurs ;
- Point commun entre toutes les machines virtuelles ;
- Hyperviseur : module KVM : (~30k SLOC)
  - `/dev/kvm` : interface `ioctl` ;
  - 2 hypercalls : `KVM_HC_VAPIC_POLL_IRQ` ( $\Leftrightarrow$  `VM_EXIT`) et `KVM_HC_KICK_CPU`.
- Drivers spécifiques :
  - `vhost-net` ;
  - `vhost-scsi`.

220/281

## Durcissement KVM

Éviter :

- les drivers dans le noyau : pas de `vhost-net/scsi` ;
- les émulateurs matériels dans le noyau : pas de PIC, APIC, IOAPIC :
  - À désactiver avec des options pour QEMU [42,43].

221/281

## Éviter Xen

- De très nombreuses vulnérabilités ;
- Dernier Cloud provider majeur : Amazon AWS.
- Migration en cours vers KVM (Nitro).

222/281

## Noyau Linux : durcissement

- Cf. chapitre sur les conteneurs.

223/281



## Système d'exploitation

- Le système d'exploitation hôte de l'hyperviseur est un système classique ;
- Cf. chapitre sur les conteneurs.

224 / 281

## Gestionnaire de machines virtuelles (VMM)

- Le gestionnaire dispose de tous les accès ;
- Contrôle d'accès ;
- Exemple pour libvirt :
  - Accès à la socket libvirt en local ;
  - Accès à distance en TCP.

225 / 281

## Emulateur / virtualiseur

- Émulateur : QEMU (~500k SLOC) ;
- Process en espace utilisateur :
  - Réduction de la surface d'attaque : moins d'émulateurs ;
  - Options de compilation / durcissement ;
  - Utilisation de seccomp-bpf ;
  - Confinement avec SELinux.

226 / 281

## Alternatives à QEMU ?

- lkvm / Native KVM tool [52] :
  - virtualisateur seulement ;
  - supporte seulement les périphériques virtio & guests Linux.
- crosvm (Chrome OS) [51] & Firecracker (AWS) :
  - écrit presque exclusivement en Rust ;
  - virtualisateur x64 seulement ;
  - supporte seulement certains périphériques virtio.
- novm (Google) :
  - écrit en Go ;
  - supporte seulement certains périphériques virtio.
- Google Compute Engine [53] ;
- Amazon Web Services Nitro [54] ;
- NEMU (Intel) : version nettoyée de QEMU.

227 / 281

## Isolation entre les machines virtuelles

- SELinux : confinement des instances de QEMU (sVirt) ;
- Voir chapitre sur la sécurité des conteneurs ;
- Catégories associées aux images disques des machines virtuelles.

228/281

## Sécurité des machines virtuelles

- Système d'exploitation complet ;
- Contraintes similaires : mise à jour, durcissement standard, etc.
- Voir chapitre sur la sécurité des conteneurs.

229/281

## Sécurité des machines virtuelles : `root`

- Protection du compte `root` et du noyau impérative ;
- La quasi totalité des vulnérabilités dans les hyperviseurs nécessitent d'être `root` (Ring 0) dans la machine virtuelle ;
- Séparation de privilèges et défense en profondeur nécessaire.

230/281

## Entropie dans une machine virtuelle

- Beaucoup de composants matériels émulés dans une VM ;
- Comportement très peu aléatoire ;
- Entropie de mauvaise qualité ;
- Solution : VirtIO-RNG.

231/281

## Images de machines virtuelles

- Vérifier l'origine et l'intégrité des images de machines virtuelles ;
- Considérer les images de machines virtuelles comme publiques ;
- Cloud-init & Ignition : configuration spécifique à chaque instance ;
- Voir chapitre sur la sécurité des conteneurs pour la gestion des secrets.

232/281

## Plan

- 1 Infrastructures
- 2 Conteneurs
- 3 systemd
- 4 Gestionnaires de conteneurs
- 5 SELinux
- 6 Conteneurs et sécurité
- 7 Virtualisation et sécurité
- 8 Sécurité des infrastructures
  - Openstack
  - Kubernetes
  - Sécurité

233/281

## Automatisation / Orchestration

- Gestion automatique des applications ;
- Gestion automatique des ressources ;
- Gestion automatique de la disponibilité.
  
- Objectif : déléguer les tâches ingrates à des services ;
- Minimiser les temps de panne et augmenter la réactivité.

234/281

## Plan

- 8 Sécurité des infrastructures
  - Openstack
  - Kubernetes
  - Sécurité

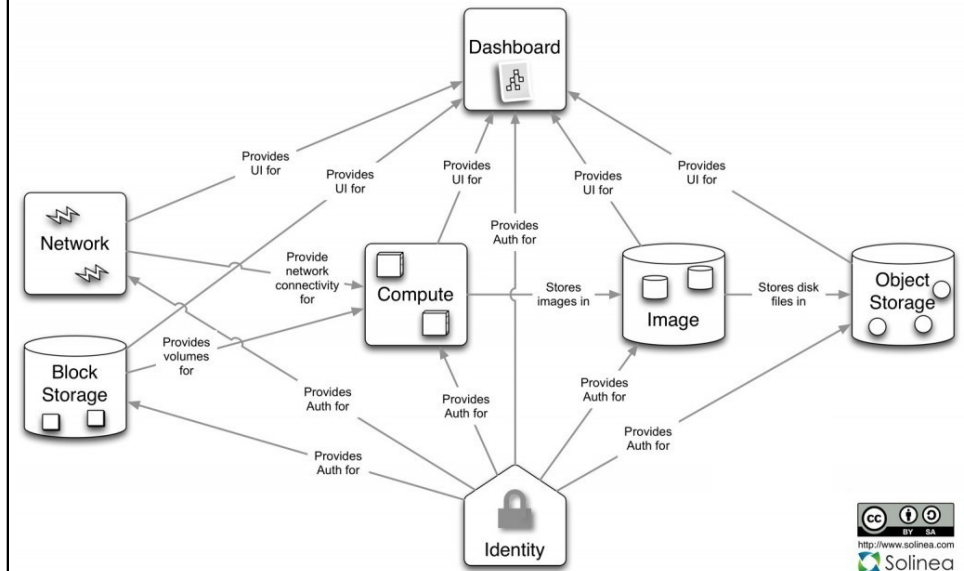
235/281

# OpenStack

- Projet lancé en 2010 par Rackspace et la NASA ;
- Ensemble de composant pour gérer automatiquement le déploiement de machines virtuelles ou de conteneurs ;
- Nombreux services disponibles ;
- Intégration avec le matériel réseau, stockage, etc.

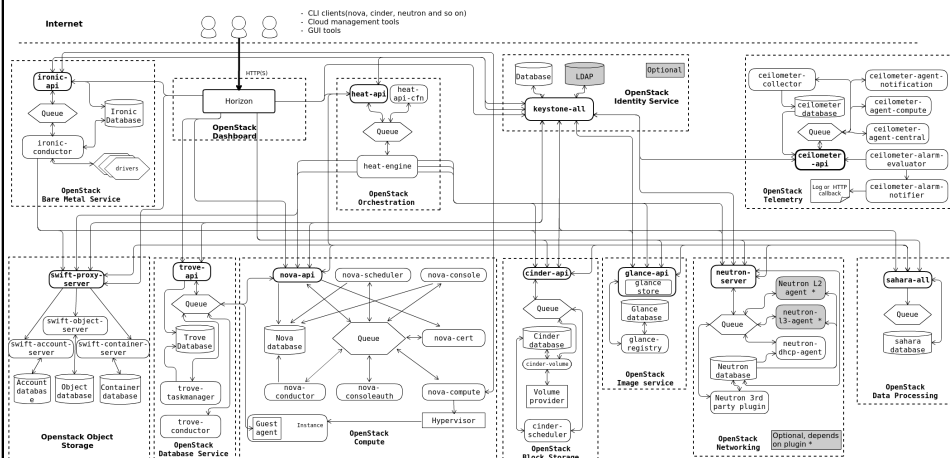
236 / 281

# OpenStack : services



237 / 281

# OpenStack : services



238 / 281

# Plan

## 8 Sécurité des infrastructures

- Openstack
- Kubernetes
- Sécurité

239 / 281

# Kubernetes

- Projet lancé par des ingénieurs chez Google en 2014 [100];
- Inspiré de *Borg*, le gestionnaire de conteneur interne Google;
- Basé sur le concept de Pods (groupe de conteneurs partageant certains namespaces [102], [103]);
- Repose sur etcd [104];
- Plusieurs distributions (OpenShift (Red Hat [107]), Typhoon, etc.).
- Regroupement de nombreux projets autour de la CNCF [101].

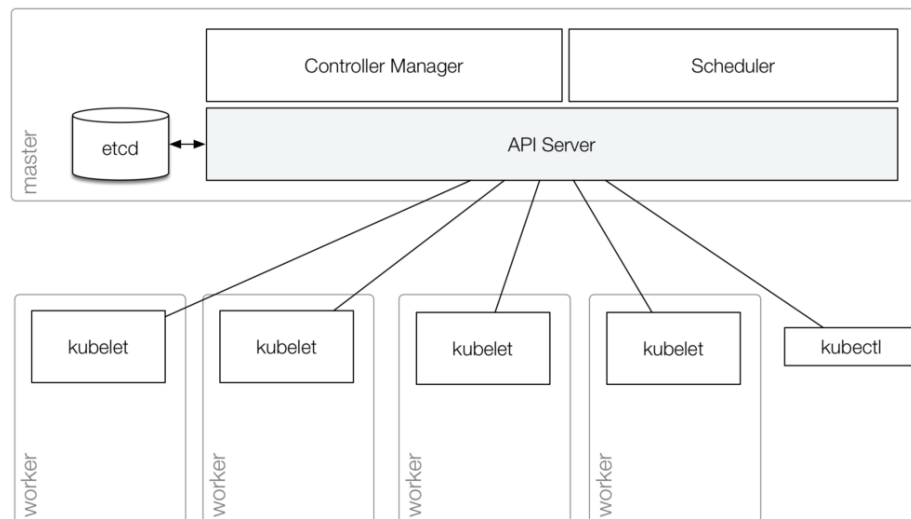
240 / 281

# Kubernetes & Cloud providers

- Versions hébergés / gérées par un Cloud provider de Kubernetes;
- Google Kubernetes Engine, Amazon EKS, Azure Kubernetes Service, etc.
- Intégré dans les offres de chaque Cloud provider.

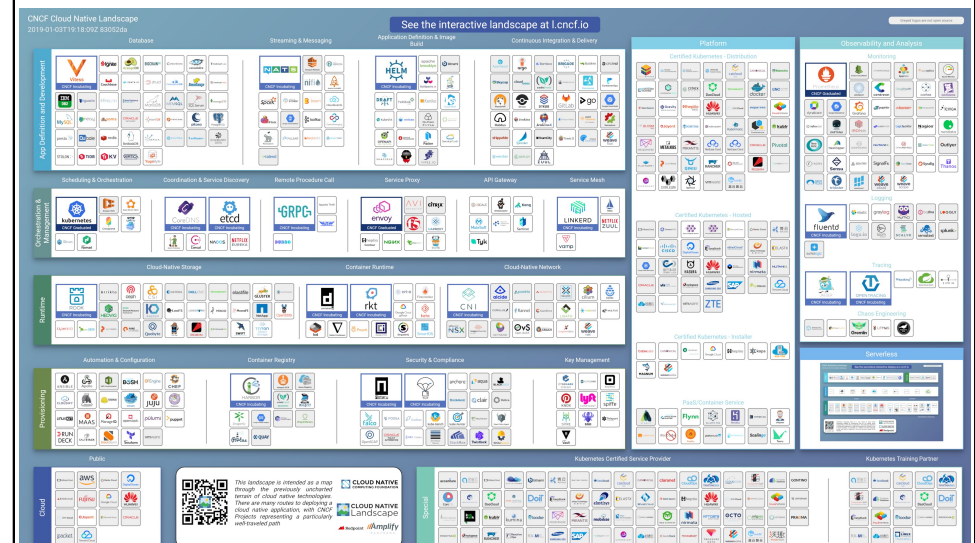
241 / 281

## Kubernetes : services



242 / 281

## Cloud Native Computing Landscape



243 / 281

## Serverless ou « Fonctions »

### Concept

Exécuter des petites portions de code en réaction à un événement et sans avoir à gérer ou mettre en place une infrastructure complète.

Exemples chez les Cloud providers :

- AWS Lambda, Google Cloud Functions, Azure Functions, Fastly Terrarium, Cloudflare Workers.

Exemples d'implémentations *open source* :

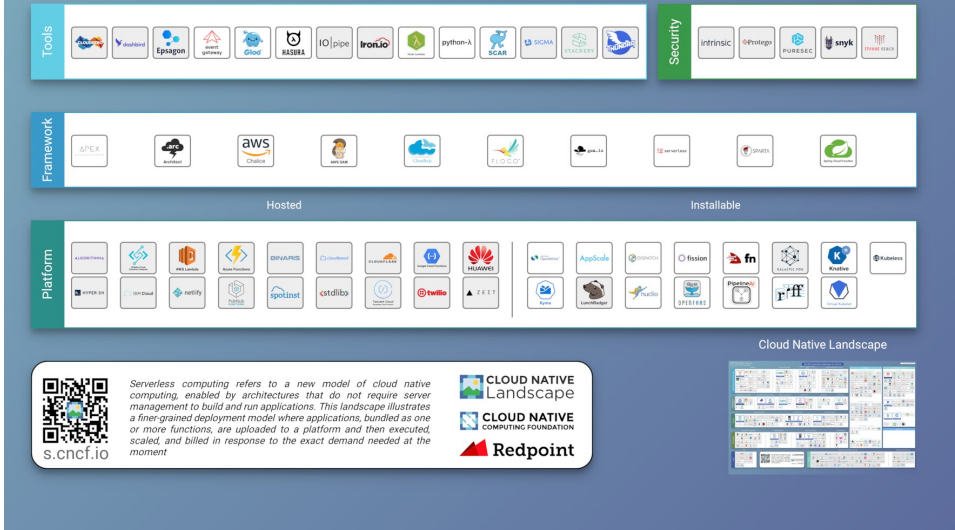
- Knative, Kubeless, Fission, Apache OpenWhisk, IronFunctions, OpenFaaS, Fn Project.

244 / 281

## Serverless

CNCF Serverless Landscape  
2019-01-03T19:18:09Z 83052da

See the serverless interactive display at [s.cncf.io](https://s.cncf.io)



245 / 281

## Plan

### 8 Sécurité des infrastructures

- Openstack
- Kubernetes
- Sécurité

246 / 281

## Sécurité des infrastructures de virtualisation

- Sécurité des éléments de base de l'infrastructure :
  - Sécurité des hôtes ;
  - Sécurité des machines virtuelles / conteneurs ;
  - Voir les chapitres précédents.
- Sécurité des services de l'infrastructure :
  - Interfaces d'administration ;
  - Interactions entre les services et gestion des réseaux ;
  - Indicateurs, visibilité et inventaire ;
  - Intégration avec les Cloud provider & matériels ;
  - Déploiement, mises à jour.

247 / 281

## Isolation des interfaces d'administration

- Isolation réseau des interfaces d'administrations des hôtes de l'infrastructure.
- Séparer les communications liées à l'administration des autres interactions ;
- Utiliser des protocoles éprouvés (SSH) ;
- Utiliser un réseau physique distinct (ou à minima un réseau logique distinct).

248/281

## Isolation des services de l'infrastructure

- La compromission d'un service peut entraîner la compromission de toute l'infrastructure [98] ;
- Séparer les communications entre les services des communications effectuées par les machines virtuelles, conteneurs et applications déployées par les utilisateurs sur l'infrastructure ;
- Utiliser des protocoles éprouvés (TLS 1.2+) ;
- Utiliser un réseau physique distinct (ou à minima un réseau logique distinct).

249/281

## Gestion du réseau

- Contrôle des interactions entre les applications / conteneurs / machines virtuelles à travers le réseau.
- OpenStack Neutron [97].
- Kubernetes :
  - Container Network Interface (CNI) ;
  - flannel (L3, vxlan) [106] ;
  - Cilium (L3-L7, HTTP, BPF) ;
  - Contiv (BGP, vxlan) ;
  - Project Calico (BGP) ;
  - etc. [105].

250/281

## Gestion des secrets & identités

- Gestion des secrets : automatiser leur création, stockage, renouvellement et expiration :
  - OpenStack : Anchor, Barbican ;
  - Kubernetes Secrets ou HashiCorp Vault.
- Gestion des identités et des accès :
  - OpenStack : Keystone ;
  - Kubernetes RBAC, Dex.

251/281

## Indicateurs, Visibilité et inventaire

- Visibilité sur l'état de l'architecture :
  - performance (Ceilometer, Prometheus, etc.);
  - audit (osquery).

252/281

## Gestion et administration

- Gestion manuelle impossible ;
- Automatisation obligatoire ;
- Les accès directs en SSH doivent être restreints aux situations critiques et au debug ;
- Déploiement automatisé (Qualification et Prod) ;
- Gestion des services à l'aide d'outils de gestion de configuration (Chef, Puppet, Ansible, etc.)
- Attention : outils non adaptés à la configuration des instances déployées par les utilisateurs sur l'infrastructure.

253/281

## Gestion et administration

### Kubernetes : Operators

- Logiciel qui se charge de gérer un déploiement dans une infrastructure Kubernetes ;
- Objectif : automatiser l'administration d'un service ;
- Exemples et état des lieux : [113].

254/281

## Mises à jour des hôtes

- La mise à jour des hôtes de l'infrastructure doit être non-interactive, atomique et coordonnées :
  - Systèmes d'exploitation centrés sur l'utilisation de conteneurs ;
  - Pas ou peu de gestion de paquets ;
  - Retour en arrière possible après une mise à jour non fonctionnelle ;
- Exemples :
  - CoreOS [90], Fedora CoreOS & Red Hat Core OS [93] ;
  - openSUSE Kubic ;
  - VMware Photon OS ;
  - Container-Optimized OS (Google Cloud seulement).

255/281



## Mise à jour de l'infrastructure ?

- Failles de sécurité inévitables. À prendre en compte ;
- Quel impact sur le fonctionnement du Cloud ?
- OpenStack : concepts d'UnderCloud et d'OverCloud (TripleO : OpenStack On OpenStack) ;
- Kubernetes : Self hosted Kubernetes (« Bootstrap » puis gestion des services de Kubernetes par lui même).
- Infrastructure as Code (Terraform) : Décrire l'infrastructure sous forme de code pour pouvoir la déployer / redéployer automatiquement.

256/281

## Cloud providers

- Pas d'accès direct à l'infrastructure ;
- Cloud provider responsable de la sécurité de l'infrastructure mise à disposition des utilisateurs ;
- Utilisateurs responsable de la bonne configuration des mécanismes de contrôle d'accès ;
- Risque principal financier !
- Protection des comptes administrateurs à l'aide de l'authentification à deux facteurs / deux étapes, avec par ordre de préférence :
  - Token hardware U2F (FIDO 1 & 2 : Yubikey, NitroKey, etc.) ;
  - TOTP hardware ou software.
- Banir l'usage du SMS.

257/281

## Sécurité OpenStack

### Ressources :

- OpenStack Security Guide [95] ;
- OpenStack Administrator Guide [96] ;
- Security and Hardening Guide (Red Hat) [99].

258/281

## Sécurité Kubernetes

- Alternative à Docker :
  - CRI-O (stabilité & sécurité).
- TLS automatique pour les communications entre conteneurs :
  - Istio, Consul, etc.
- Ressources :
  - Pod Security Policy [111] :
    - Obliger tous les conteneurs à tourner en !root ;
    - Obliger tous les conteneurs à tourner avec le / en RO.
  - Securing a Cluster [108] ;
  - Kubernetes The Hard Way [109] ;
  - Container Security Guide (Red Hat) [110] ;
  - OpenShift Container Security Guide (Red Hat) [112].

259/281

## Références I

- ① xfs\_quota(8) :  
[http://man7.org/linux/man-pages/man8/xfs\\_quota.8.html](http://man7.org/linux/man-pages/man8/xfs_quota.8.html)
- ② Control Groups Resource Management :  
<http://libvirt.org/cgroups.html>
- ③ Control Group v2 : <https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/cgroup-v2.txt>
- ④ CVE request Linux kernel : ns : user namespaces panic :  
<http://www.openwall.com/lists/oss-security/2015/05/29/5>
- ⑤ Linux kernel source: commit  
51dfcb076d1e1ce7006aa272cb7c4514740c7e47
- ⑥ Linux kernel source: commit  
87c31b39abcb6fb6bd7d111200c9627a594bf6a9

260/281

## Références II

- ⑦ Linux namespaces : It is possible to escape from bind mounts :  
<http://www.openwall.com/lists/oss-security/2015/04/03/7>
- ⑧ User namespaces + overlayfs = root privileges :  
<https://lwn.net/Articles/671641/>
- ⑨ Anatomy of a user namespaces vulnerability :  
<https://lwn.net/Articles/543273/>
- ⑩ CVE-2015-1328 : incorrect permission checks in overlayfs, ubuntu local root :  
<http://seclists.org/oss-sec/2015/q2/717>
- ⑪ /proc & /sys : Enforcing mount options for sysfs and proc : <https://lwn.net/Articles/647757/>

261/281

## Références III

- ⑫ Petit état de l'art des systèmes d'initialisation :  
<http://linuxfr.org/news/petit-etat-de-l-art-des-systemes-d-initialisation-1>
- ⑬ The Biggest Myths about systemd :  
<http://0pointer.de/blog/projects/the-biggest-myths.html>
- ⑭ Red Hat Summit 2013 - Getting Ready for systemd :  
<https://access.redhat.com/site/videos/403833>
- ⑮ systemd Freedesktop.org Wiki :  
<http://www.freedesktop.org/wiki/Software/systemd/>
- ⑯ Journal File Format : <http://www.freedesktop.org/wiki/Software/systemd/journal-files/>
- ⑰ Durcissement système à l'aide de systemd, Timothée Ravier, SSTIC 2017 : [https://www.sstic.org/2017/presentation/durcissement\\_systeme\\_avec\\_systemd/](https://www.sstic.org/2017/presentation/durcissement_systeme_avec_systemd/)

262/281

## Références IV

- ⑰ Linux Kernel Exploitation - Earning Its Pwnie a Vuln at a Time : <https://jon.oberheide.org/files/source10-linuxkernel-jonoberheide.pdf>
- ⑱ Exploiting the Linux kernel : Measures and countermeasures : <https://jon.oberheide.org/files/syscan12-exploitinglinux.pdf>
- ⑲ Dirty CoW : <https://dirtycow.ninja/>
- ⑳ rkt : <https://coreos.com/blog/rocket/>
- ㉑ Container mechanics in rkt and Linux : <http://events.linuxfoundation.org/sites/events/files/slides/Container%20mechanics%20in%20rkt%20and%20Linux.pdf>
- ㉒ rkt architecture : <https://coreos.com/rkt/docs/latest/devel/architecture.html>

263/281

## Références V

- 24 App Container - Rocket : [https://www.socallinuxexpo.org/sites/default/files/presentations/appc%20%2B%20rocket%20\(SCALE%2013x\).pdf](https://www.socallinuxexpo.org/sites/default/files/presentations/appc%20%2B%20rocket%20(SCALE%2013x).pdf)
- 25 Using Virtual Machines to Improve Container Security with rkt v0.8.0 : <https://coreos.com/blog/rkt-0.8-with-new-vm-support/>
- 26 App Container basics : <https://coreos.com/rkt/docs/latest/app-container.html>
- 27 Using rkt with systemd : <https://coreos.com/rkt/docs/latest/using-rkt-with-systemd.html>
- 28 Image Fetching Behavior : <https://coreos.com/rkt/docs/latest/image-fetching-behavior.html>
- 29 rkt vs other projects : <https://coreos.com/rkt/docs/latest/rkt-vs-other-projects.html>

264/281

## Références VI

- 30 LXC CVE-2013-6441 : <http://www.cvedetails.com/cve/CVE-2013-6441/>
- 31 Security issue in LXC (CVE-2015-1335) : <http://www.openwall.com/lists/oss-security/2015/09/29/4>
- 32 Before you initiate a “docker pull” : <https://securityblog.redhat.com/2014/12/18/before-you-initiate-a-docker-pull/>
- 33 Docker 1.8 : <https://blog.docker.com/2015/08/content-trust-docker-1-8/>
- 34 The Update Framework (TUF) : <https://theupdateframework.github.io/>
- 35 Docker Registry : <https://github.com/docker/distribution>

265/281

## Références VII

- 36 Over 30% of Official Images in Docker Hub Contain High Priority Security Vulnerabilities : <http://www.banyanops.com/blog/analyzing-docker-hub/>
- 37 Docker 1.3.2 - Security Advisory [24 Nov 2014] : <http://www.openwall.com/lists/oss-security/2014/11/24/5>
- 38 Docker 1.3.3 - Security Advisory [11 Dec 2014] : <https://groups.google.com/forum/#!msg/docker-user/nFAz-B-n4Bw/0wr3wvLsnUwJ>
- 39 Docker 1.6.1 - Security Advisory [150507] : <http://seclists.org/fulldisclosure/2015/May/28>
- 40 False Boundaries and Arbitrary Code Execution : <https://forums.grsecurity.net/viewtopic.php?f=7&t=2522>
- 41 CAP\_SYS\_ADMIN : the new root : <https://lwn.net/Articles/486306/>

266/281

## Références VIII

- 42 KVM Security Improvements, Andrew Honig, KVM Forum 2014 : <http://www.linux-kvm.org/images/f/f6/01x02-KVMHardening.pdf>
- 43 Performant Security Hardening, Steve Rutherford, KVM Forum 2016 : [https://www.linux-kvm.org/images/3/3d/01x02-Steve\\_Rutherford-Performant\\_Security\\_Hardening\\_of\\_KVM.pdf](https://www.linux-kvm.org/images/3/3d/01x02-Steve_Rutherford-Performant_Security_Hardening_of_KVM.pdf)
- 44 XDC2012 : Graphics stack security : <https://lwn.net/Articles/517375/>
- 45 Reverse Engineering Intel DRAM Addressing and Exploitation : <http://arxiv.org/abs/1511.08756>
- 46 Cross Processor Cache Attacks : <http://eprint.iacr.org/2015/1155>

267/281

## Références IX

- 47 RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis : <http://arstechnica.com/security/2013/12/new-attack-steals-e-mail-decryption-keys-by-capturing-computer-sounds/>
- 48 Introducing Blue Pill : <http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html>
- 49 Virtunoid : A KVM Guest → Host privilege escalation exploit : [http://media.blackhat.com/bh-us-11/Elhage/BH\\_US\\_11\\_Elhage\\_Virtunoid\\_WP.pdf](http://media.blackhat.com/bh-us-11/Elhage/BH_US_11_Elhage_Virtunoid_WP.pdf)
- 50 VENOM, don't get bitten : <https://securityblog.redhat.com/2015/05/13/venom-dont-get-bitten/>
- 51 crosvm - The Chrome OS Virtual Machine Monitor : <https://chromium.googlesource.com/chromiumos/platform/crosvm/>

268/281

## Références X

- 52 Stand-alone KVM tool : <https://git.kernel.org/pub/scm/linux/kernel/git/will/kvmtool.git>
- 53 7 ways we harden our KVM hypervisor at Google Cloud : security in plaintext : <https://cloudplatform.googleblog.com/2017/01/7-ways-we-harden-our-KVM-hypervisor-at-Google-Cloud-security-in-plaintext.html>
- 54 AWS EC2 Virtualization 2017 : Introducing Nitro : <http://www.brendangregg.com/blog/2017-11-29/aws-ec2-virtualization-2017.html>
- 55 Xen Security Advisory CVE-2014-8594 / XSA-109 : <http://xenbits.xen.org/xsa/advisory-109.html>
- 56 Xen Security Advisory CVE-2015-2151 / XSA-123 : <http://xenbits.xen.org/xsa/advisory-123.html>

269/281

## Références XI

- 57 Xen Security Advisory CVE-2015-7835 / XSA-148 : <http://xenbits.xen.org/xsa/advisory-148.html>
- 58 Qubes Security Bulletin #22 : <https://github.com/QubesOS/qubes-secpack/blob/master/QSBs/qsb-022-2015.txt>
- 59 Xen Security Advisory CVE-2015-8550 / XSA-155 : <http://xenbits.xen.org/xsa/advisory-155.html>
- 60 Meltdown and Spectre : <https://meltdownattack.com/>
- 61 The Intel SYSRET privilege escalation : <http://blog.xen.org/index.php/2012/06/13/the-intel-sysret-privilege-escalation/>
- 62 Attacking SMM Memory via Intel CPU Cache Poisoning : <http://theinvisiblethings.blogspot.fr/2009/03/attacking-smm-memory-via-intel-cpu.html>

270/281

## Références XII

- 63 KVM SYSENTER emulation vulnerability - CVE-2015-0239 : <http://www.openwall.com/lists/oss-security/2015/01/27/6>
- 64 Linux kernel source: commit [a08d3b3b99efd509133946056531cdf8f3a0c09b](https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=a08d3b3b99efd509133946056531cdf8f3a0c09b)
- 65 CVE-2011-4127 : privilege escalation from qemu / KVM guests : <http://rwmj.wordpress.com/2011/12/22/cve-2011-4127-privilege-escalation-from-qemu-kvm-guests/>
- 66 Network Function Virtualization, Packet Processing Performance of Virtualized Platforms with Linux and Intel Architecture : [https://networkbuilders.intel.com/docs/network\\_builders\\_RA\\_NFV.pdf](https://networkbuilders.intel.com/docs/network_builders_RA_NFV.pdf)

271/281

## Références XIII

- 67 A reminder why you should never mount guest disk images on the host OS :  
<https://www.berrange.com/posts/2013/02/20/a-reminder-why-you-should-never-mount-guest-disk-images-on-the-host-os/>
- 68 FUSE : Filesystem in Userspace :  
[http://en.wikipedia.org/wiki/Filesystem\\_in\\_Userspace](http://en.wikipedia.org/wiki/Filesystem_in_Userspace)
- 69 Problématiques de sécurité associées à la virtualisation des systèmes d'information :  
<http://www.ssi.gouv.fr/guide/problematiques-de-securite-associees-a-la-virtualisation-des-systemes-din>
- 70 Recommandations de sécurité relatives à un système GNU/Linux : <http://www.ssi.gouv.fr/guide/recommandations-de-securite-relatives-a-un-systeme-gnulinux/>

272/281

## Références XIV

- 71 Recommandations de configuration matérielle de postes clients et serveurs x86 :  
<http://www.ssi.gouv.fr/administration/guide/recommandations-de-configuration-materielle-de-postes-clients-et-s>
- 72 Recommandations relatives à l'administration sécurisée des systèmes d'information :  
<http://www.ssi.gouv.fr/administration/guide/securiser-ladministration-des-systemes-dinformation/>
- 73 Recommandations de sécurité pour la mise en oeuvre d'un système de journalisation :  
<http://www.ssi.gouv.fr/administration/guide/recommandations-de-securite-pour-la-mise-en-oeuvre-dun-systeme->
- 74 NIST Application Container Security Guide :  
<https://www.nist.gov/publications/application-container-security-guide>

273/281

## Références XV

- 75 Security Assurance Requirements for Linux Application Container Deployments :  
<https://csrc.nist.gov/publications/detail/nistir/8176/final>
- 76 NIST Guidance on Application Container Security :  
<https://csrc.nist.gov/publications/detail/itl-bulletin/2017/10/application-container-security/final>
- 77 Container Security with SELinux and CoreOS : <https://coreos.com/blog/container-security-selinux-coreos.html>
- 78 Announcing rkt v0.7.0, featuring a new build system, SELinux and more : <https://coreos.com/blog/rkt-0.7.0-with-selinux-and-new-build-system.html>

274/281

## Références XVI

- 79 Docker SELinux Security Policy :  
<https://access.redhat.com/documentation/en/red-hat-enterprise-linux-atomic-host/7/container-security-guide/chapter-6-docker-selinux-security-policy>
- 80 State of SELinux, Paul Moore, LSS 2017 :  
[http://www.paul-moore.com/docs/lss-state\\_of\\_selinux-pmoore-092017-r3.pdf](http://www.paul-moore.com/docs/lss-state_of_selinux-pmoore-092017-r3.pdf)
- 81 Honey, I Shrunk the Attack Surface, Adventures in Android Security Hardening, Nick Kravich, Black Hat 2017 : <https://www.blackhat.com/docs/us-17/thursday/us-17-Kravich-Honey-I-Shrunk-The-Attack-Surface-Adventures-In-A> pdf

275/281

## Références XVII

- 82 What roles do DAC (file permissions), ACL and MAC (SELinux) play in Linux file security ? : <https://unix.stackexchange.com/questions/16828/what-roles-do-dac-file-permissions-acl-and-mac-selinux-play-in-linux-file-s>
- 83 Where does CIL play in the SELinux system ? : <http://blog.siphos.be/2015/06/where-does-cil-play-in-the-selinux-system/>
- 84 Docker Security : Using Containers Safely in Production : <https://www.openshift.com/promotions/docker-security.html>
- 85 Docker security : <https://docs.docker.com/engine/articles/security/>

276/281

## Références XVIII

- 86 Why we don't let non-root users run Docker in CentOS, Fedora, or RHEL : <http://www.projectatomic.io/blog/2015/08/why-we-dont-let-non-root-users-run-docker-in-centos-fedora-or-rhel/>
- 87 Containers & Docker : How Secure Are They ? : <http://blog.docker.io/2013/08/containers-docker-how-secure-are-they/>
- 88 HashiCorp Vault : <https://www.vaultproject.io>
- 89 Remote Code Execution in Alpine Linux : <https://justi.cz/security/2018/09/13/alpine-apk-rce.html>
- 90 CoreOS : <https://coreos.com/>
- 91 What Trusted Computing Means to Users of CoreOS and Beyond : <https://coreos.com/blog/coreos-trusted-computing.html>

277/281

## Références XIX

- 92 Making Sense of Container Standards and Foundations : OCI, CNCF, appc and rkt : <https://coreos.com/blog/making-sense-of-standards.html>
- 93 Project Atomic : <http://www.projectatomic.io/>
- 94 Docker Compose : <https://docs.docker.com/compose/>
- 95 OpenStack Security Guide : <https://docs.openstack.org/security-guide/>
- 96 OpenStack Administrator Guides : <https://docs.openstack.org/admin/>
- 97 There's Real Magic behind OpenStack Neutron : <https://pinrojas.com/2014/07/29/theres-real-magic-behind-openstack-neutron/>

278/281

## Références XX

- 98 OpenStack logical architecture : <https://docs.openstack.org/install-guide/get-started-logical-architecture.html>
- 99 Security and Hardening Guide, Red Hat : [https://access.redhat.com/documentation/en-us/red\\_hat\\_openstack\\_platform/12/html/security\\_and\\_hardening\\_guide/](https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/12/html/security_and_hardening_guide/)
- 100 Kubernetes : <http://kubernetes.io/>
- 101 Cloud Native Computing Foundation : <https://www.cncf.io/>
- 102 Kubernetes user guide - Pods : <http://kubernetes.io/v1.1/docs/user-guide/pods.html>
- 103 App Container Spec - Pods : <https://github.com/appc/spec/blob/master/spec/pods.md#app-container-pods-pods>
- 104 etcd : <https://coreos.com/etcd/>

279/281

## Références XXI

- ① Kubernetes Cluster Networking : <https://kubernetes.io/docs/concepts/cluster-administration/networking/>
- ① flannel : <https://coreos.com/flannel/docs/latest/>
- ① OpenShift Origin : <https://www.openshift.org/>
- ① Kubernetes - Securing a Cluster : <https://kubernetes.io/docs/tasks/administer-cluster/securing-a-cluster/>
- ① Kubernetes The Hard Way, Kelsey Hightower : <https://github.com/kelseyhightower/kubernetes-the-hard-way>
- ① Container Security Guide, Red Hat : [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux\\_atomic\\_host/7/html-single/container\\_security\\_guide/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux_atomic_host/7/html-single/container_security_guide/index)

280/281

## Références XXII

- ① Pod Security Policy : <https://kubernetes.io/docs/concepts/policy/pod-security-policy/>
- ① OpenShift Container Security Guide : <https://docs.openshift.com/container-platform/3.7/security/index.html>
- ① Kubernetes Operators : <https://kubedex.com/operators/>
- ① CoreOS Introduces Clair : Open Source Vulnerability Analysis for your Containers : <https://coreos.com/blog/vulnerability-analysis-for-containers/>
- ① Security compliance of RHEL7 Docker containers : <https://www.open-scap.org/resources/documentation/security-compliance-of-rhel7-docker-containers/>

281/281