Sécurité des systèmes d'exploitation

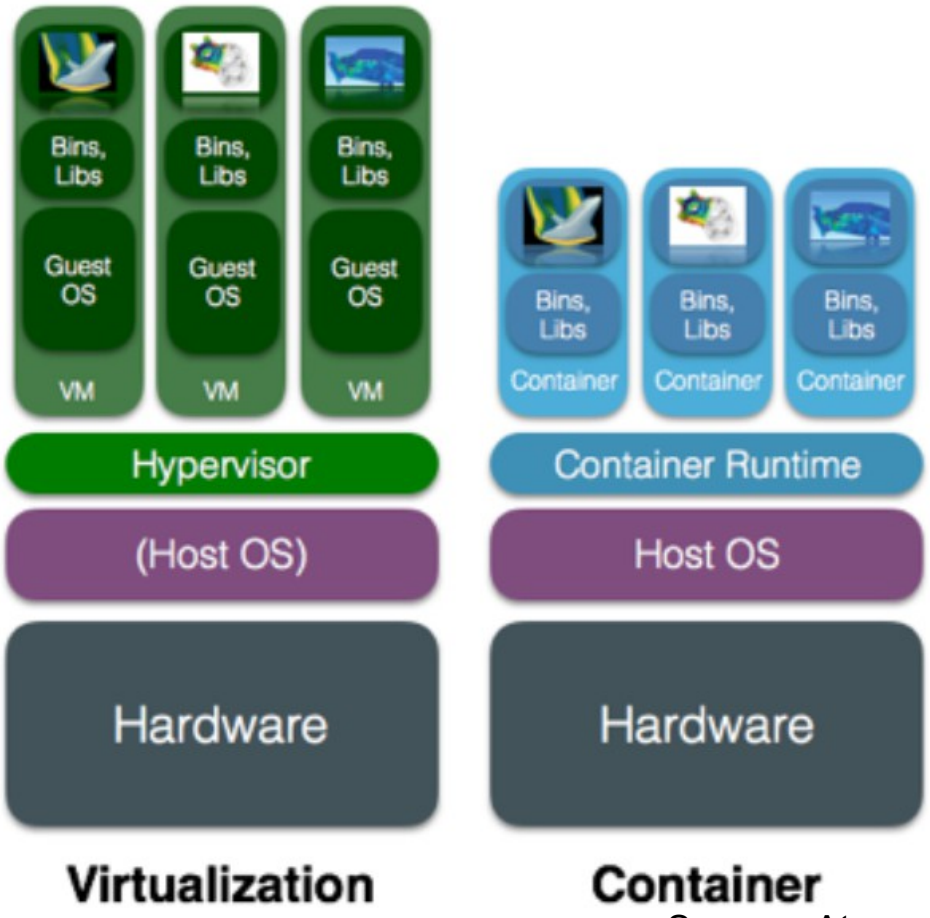
Conteneurs

Plan

- Définition
- Mécanismes techniques
- Sécurité
- Implémentation Windows

Définition

- Parfois appelés virtualisation légère ou *OS-Virtualization*
  - *Linux Containers* ou LXC (2008), Docker (2013)
  - Pas spécifique à Linux : FreeBSD *Jails* (2000), *Zones* Solaris (2005)
- Partagent un même noyau
- Reposent sur des mécanismes d'isolation offerts par le noyau



Source : Atos

Mécanismes techniques

- Conteneurs Linux reposent généralement sur
  - 4 composants principaux
    - Les espaces de noms (*namespaces*) qui présentent des « vues » différentes des ressources du système (réseau, processus, système de fichiers, utilisateurs, …)
    - Les cgroups (*Control Groups*) qui limitent les ressources matérielles (processeur, E/S disque, mémoire, ...)
    - Les capacités Linux permettent de réduire les privilèges du compte root dans les conteneurs
    - L'appel système pivot_root() qui bascule dans l'environnement du conteneur en changeant la racine du système fichiers
  - composants optionnels (pas pour la sécurité !) : le contrôle d'accès obligatoire (AppArmor ou SELinux), politique SECCOMP

Mécanismes techniques

- Espaces de noms
  - Repose sur l'idée commune d'un découpage logique
  - S'applique à un processus ou un groupe de processus (i.e., un conteneur)
  - Créés à partir de l'appel système clone() en utilisant les drapeaux CLONE_NEW* lors de la création d'un processus
    - 2 autres appels système pour manipuler les espaces de noms
      - setns() : permet de rejoindre un espace de noms ciblés par un descripteur de fichier de type /proc/<pid>/ns
      - unshare() déplace le processus appelant vers un nouvel espace de nom

Mécanismes techniques

- Types disponibles
  - PID (CLONE_NEWPID) : fournit au nouveau processus créé une vue vierge de tout processus existant en lui attribuant le PID 1
  - *Mount* (CLONE_NEWNS) : apporte une vue spécifique des systèmes de fichiers montés
    - Peut sécuriser indirectement d'autres espaces de noms en limitant l'accès à l'instance /proc de l'hôte (par ex., vue des processus de l'hôte)
  - IPC (CLONE_NEWIPC) : concerne les IPC System V et les files de messages POSIX
    - Par ex., les segments de mémoire partagée

Mécanismes techniques

- *UTS* (CLONE_NEWUTS) : sépare les noms d'hôte et de domaine
  - Utile pour donner une « identité » différente à un conteneur (par ex., dans les journaux)
- *Network* (CLONE_NEWNET) : créé un nouvel environnement réseau avec des instances différentes des interfaces réseau :
  - piles protocolaires IPv4 et IPv6, tables de routage IP, règles de pare-feu, arborescence /proc/net, sockets, …
    - Une interface réseau physique ne peut être présente que dans un seul espace de noms, l'accès à cette interface depuis d'autres espaces de noms peut se faire via une interface virtuel

Mécanismes techniques

- *User* (CLONE_NEWUSER) : permet de faire croire à un processus qu'il s'exécute en tant que root (uid 0) au sein de l'espace de noms alors qu'à l'extérieur de celui, l'identifiant est celui d'un utilisateur non-privilégié
  - Utilisation d'une table de correspondance entre les UID de l'hôte et du conteneur
  - Fournit une barrière de sécurité primordiale en réduisant la surface d'attaque du noyau
  - L'exécution d'un binaire « setuid root », ou doté de toutes les capacités, au sein de l'espace de noms ne permet pas, par construction, d'utiliser ces privilèges à l'extérieur de l'espace de noms
    - Facilite la création de conteneur sans privilèges
    - N'exclue pas des défauts d'implémentation ...

Mécanismes techniques

- Commencée depuis la version 2.4 du noyau, l'évolution des espaces de noms n'est pas terminée
  - Intégration progressive car ne fait pas partie de la conception initiale => source de bugs et de vulnérabilités
  - Propositions pour les périphériques, le temps, syslog, les pseudo-systèmes de fichiers proc et sys, ...
  - Cas particulier des LSM « lourds »
    - L'hôte fournit AppArmor, un conteneur utilise SELinux pour sa sécurité => peut-on faire cohabiter les 2 ?
      - Pas possible avec le noyau Linux actuel (4.14)
        - Espace de noms dédié à l'avenir ?

Mécanismes techniques

- Control groups
  - Partitionnement de l'ensemble des processus ou threads en des sous-ensembles (les « groupes ») sur lesquels peuvent s'appliquer différents contrôles, à vocation de performance ou de sécurité
    - Contrôles réalisés par des sous-systèmes du noyau appelé contrôleurs de ressources
  - Configurables via /sys/fs/cgroup (pseudo-système de fichiers) ou avec des outils comme cgcreate, cgexec ou cgclassify
  - Il existe un type d'espace de noms pour les Cgroups (CLONE_NEWCGROUP) : créé une vue pour un processus limitée à ses Cgroups

Mécanismes techniques

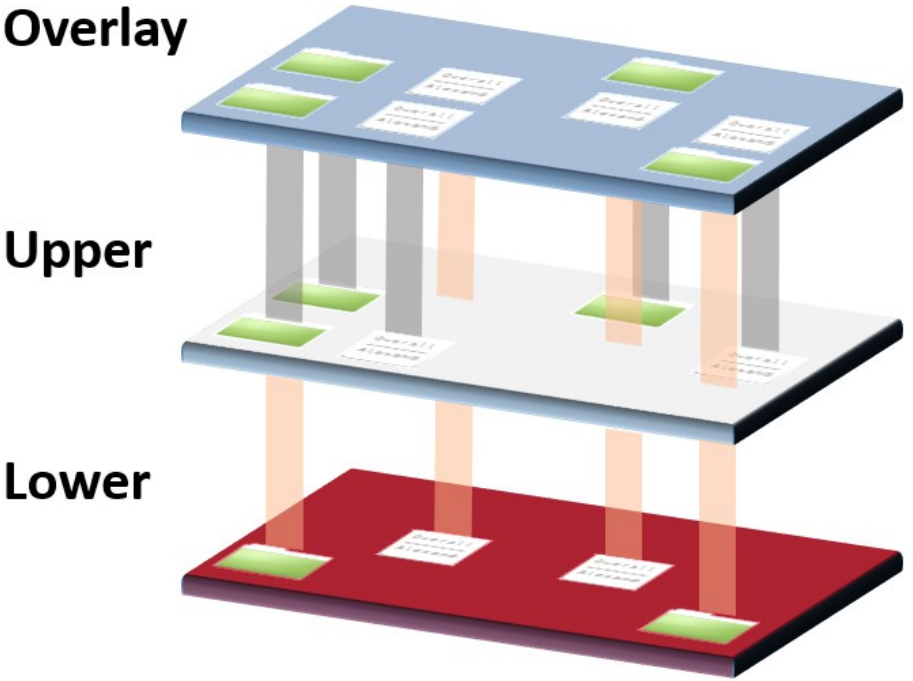
- Contrôleurs majeurs
  - CPU (cpu, cpuset, cpuacct) : utilisé pour restreindre un ensemble de processus à un nombre spécifique de processeur ou de temps processeur
  - *Memory* : contrôle l'allocation mémoire et ses limites d'un ensemble de processus
    - Les limites peuvent être figées, souples ou dynamiques (fonction de l'activité machine)
  - *Network* (net_cls) : permet d'étiqueter les paquets réseaux avec une valeur « classid », utilisable dans les règles iptables (filtrage de paquet) ou de la QoS
  - *Freezer* : mettre en pause (« geler ») un ensemble de processus via un signal SIGSTOP ou de les réveiller (« dégeler ») via un signal SIGCONT

Mécanismes techniques

- *Devices* : impose des restrictions sur l'accès aux périphériques à un ensemble de processus
  - Une approche en liste blanche est possible
    - Contenu courant de cette liste : /dev/null et /dev/urandom (mais pas /dev/random)
- BLKIO : mise en place, pour un ensemble de processus, d'une politique d'accès (vitesse de lecture/écrite disque, opérations par seconde, contrôles des files, temps d'attente, …) aux périphériques de type bloc
- PID : limitation du nombre de processus
  - Prévention des « fork-bombs » intentionnelles ou accidentelles

Mécanismes techniques

- Utilisation de systèmes de fichiers en couche
  - Partage d'une base commune issue de l'hôte
  - Seules les modifications sont enregistrées (via *Copy on Write*) sur la couche la plus haute
  - AUFS (dérivé de UnionFS), OverlayFS



Source : Datalight

Sécurité

- Les risques liés aux conteneurs sont très liés au noyau Linux
  - Maturité de l'implémentation des espaces de noms
  - Gestion des capacités et des privilèges du compte root
  - Noyau partagé par tous les conteneurs
    - Présente généralement une surface d'attaque importante
    - Maîtrise des vulnérabilités du noyau ?
- Solution 100 % logicielle (pas de recours à des mécanismes d'isolation apportés par le matériel)
  - Gain en flexibilité mais risque plus élevé si des périphériques sont directement exposés

Sécurité

- Menaces applicables
  - Évasion (cf. virtualisation et évasion de l'invité vers l'hôte), reposant sur :
    - Absence d'espaces de noms
    - Configuration par défaut non-sécurisée
      - Persistance de capacités non-indispensables
    - Exposition de l'hôte (via procfs et sysfs) => obtention d'informations utiles pour un attaquant ou reconfiguration de l'hôte en cas de mauvaise protection
      - Modification des variables noyau /proc/sys/ via sysctl()
      - Configuration du noyau (via /proc/config.gz si le paramètre CONFIG_IKCONFIG_PROC a été activé)
      - /proc/sys/security (securityfs) : accès à la configuration de LSM comme AppArmor ou SELinux
      - /sys/firmware/efi/vars : interaction avec les variables (U)EFI
    - Interface réseau partagée avec l'hôte

Sécurité

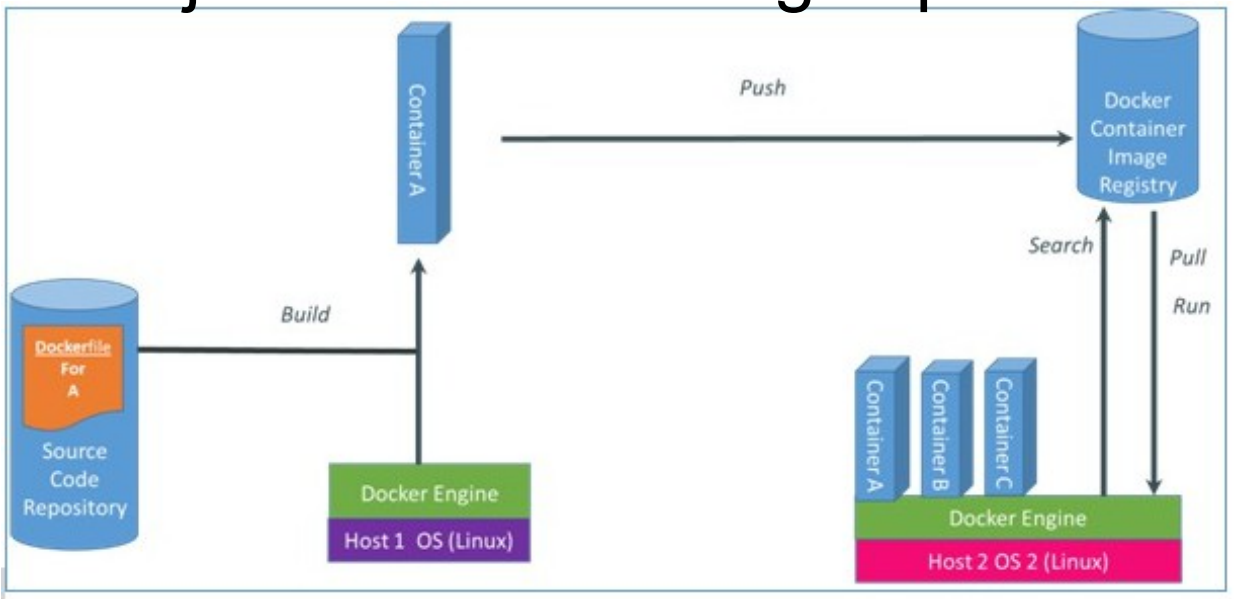
- Attaques entre conteneurs
- Attaques ciblant le conteneur
  - Logiciels non-mis à jour ou développés sans prise en compte de la sécurité
  - Exposition du conteneur à des réseaux potentiellement hostiles
  - Utilisation d'images avec une base logicielle élargie
    - Présence d'outils facilitant les actions de l'attaquant (par ex., interpréteur Python) ou son accès distant (par ex., ssh ou netcat)
  - Déni de service et consommation excessive de ressources
    - Processeur, mémoire, E/S stockage
    - Génération d'aléa : un conteneur peut assécher les réservoirs d'entropies du noyau via /dev/random ou l'appel système getrandom()

Sécurité

- Attaques classiques mais renouvelées
  - Attaques sur la couche de gestion
    - Plusieurs attaques ont été recensées sur Docker et LXC :
      - L'accès à procfs (mauvaise gestion du montage)
      - Gestion des liens (symbolique ou physique)
      - Attaque de type *Directory traversal*
      - Fichiers sensibles accessibles en écriture
  - Attaques avancées ciblant du matériel
  - Utilisation de code non éprouvé
    - Bug inhérents à la production de code source => peuvent se transformer en vulnérabilités
    - Défauts de conceptions ou d'implémentation
    - Amplifié par le mode d'utilisation des conteneurs
      - Devops, Software Defined *, ...

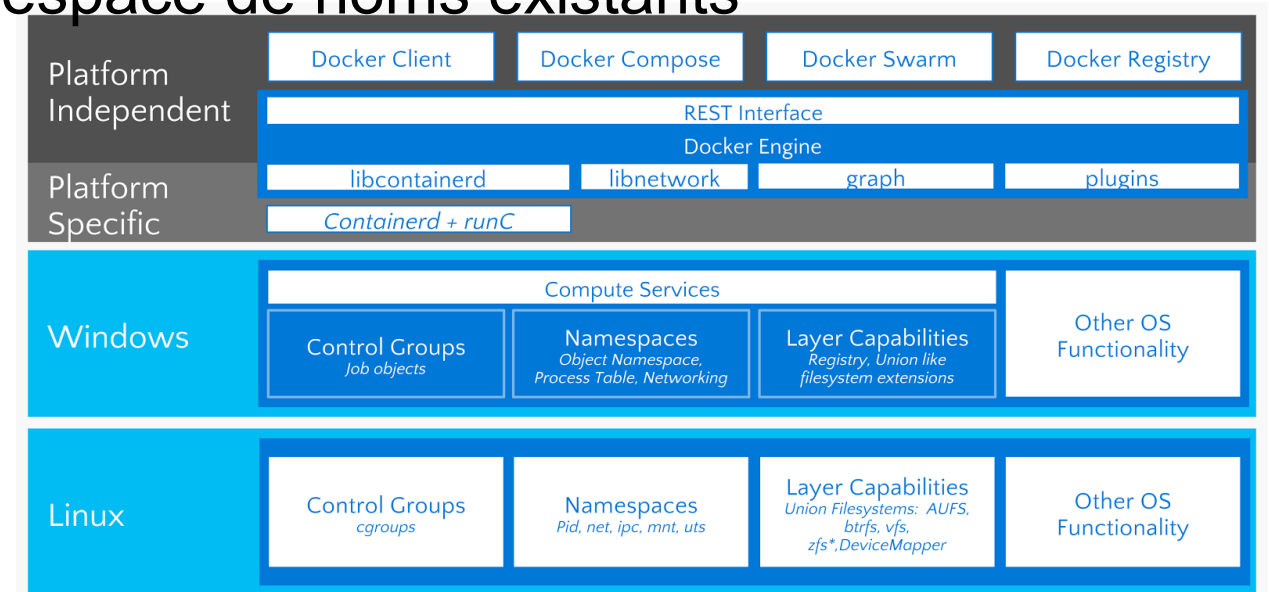
Sécurité

- Attaques indirectes
  - Concentration du code des outils sur des dépôts comme GitHub
    - Réaction en cas de compromission de GitHub ?
  - Images malveillantes ou compromises
    - Contrôle du contenu, mécanismes d'attestation ?
  - Gestions des mises à jour dans des images produites par des tiers



Implémentation Windows

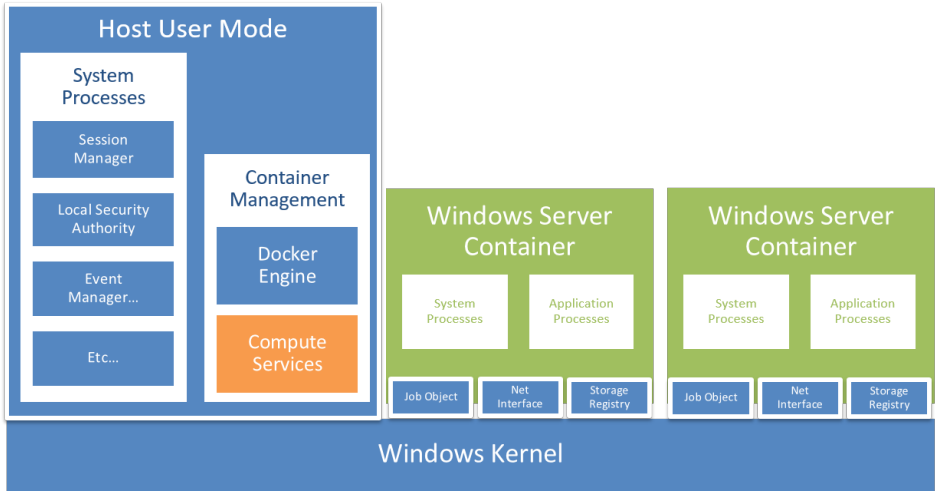
- Intégration récente (Windows 10 1607 & Windows Server 2016) des conteneurs via un portage du moteur Docker
  - Docker est construit sur des mécanismes inhérents au noyau Linux
    - Introduction d'un nouvel objet Silo (extension des objets de type Job) intégrant une extension des mécanismes d'espace de noms existants

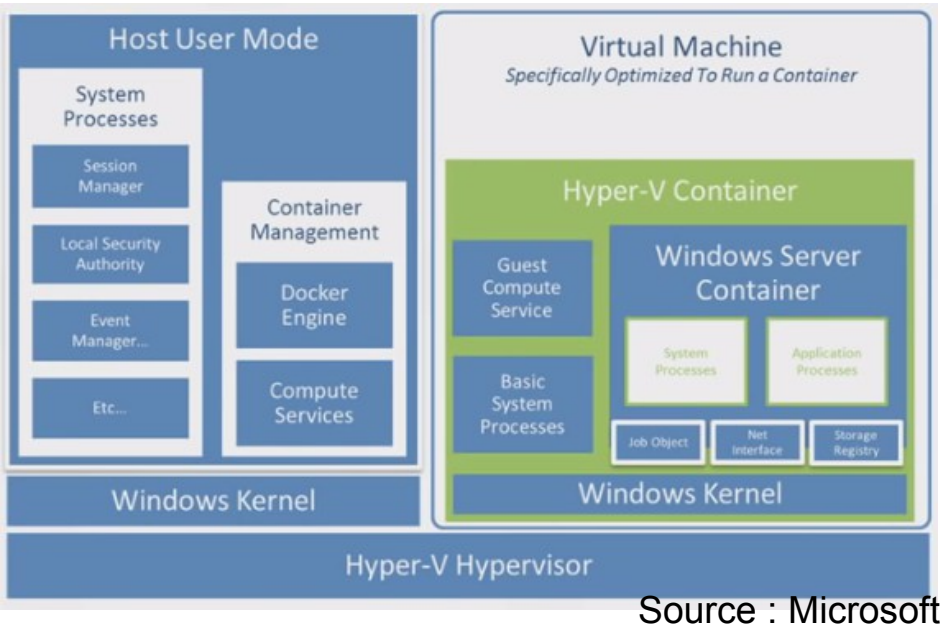


Source : Microsoft

Implémentation Windows

- 2 approches :
  - Windows Server Containers
    - Utilisation des nouveaux mécanismes du noyau
      - Sécurité ?
    - Ne fonctionne qu'avec des images construit sous Windows
  - Hyper-V Containers
    - Un conteneur = une VM
    - Utilisation de VM Linux pour héberger des images construit sous Linux





Source : Microsoft

Pour aller plus loin

- Understanding and Hardening Linux Containers, Aaron Grattafiori (NCC Group), 2016
- Abusing Privileged and Unprivileged Linux Containers, Jesse Hertz (NCC Group), 2016
- Docker : quelle sécurité pour les conteneurs, MISC n°95, 2018

Questions ?