SÉCURITÉ DES CONTENEURS LINUX

SECURIMAG - 28 NOV. 2019

0hexit

QU'EST-CE QU'UN CONTENEUR ?

- **Déf. 1 :** *~ chroot on steroids*
- **Déf. 2 :** ensemble de processus ayant les mêmes "étiquettes" namespaces et cgroups

VRAI OU FAUX (OU UN PEU DES DEUX) ?

- Affirmation 1 :

> *Les conteneurs n'existent pas*
> *(Jean-Tiare Le Bigot)*

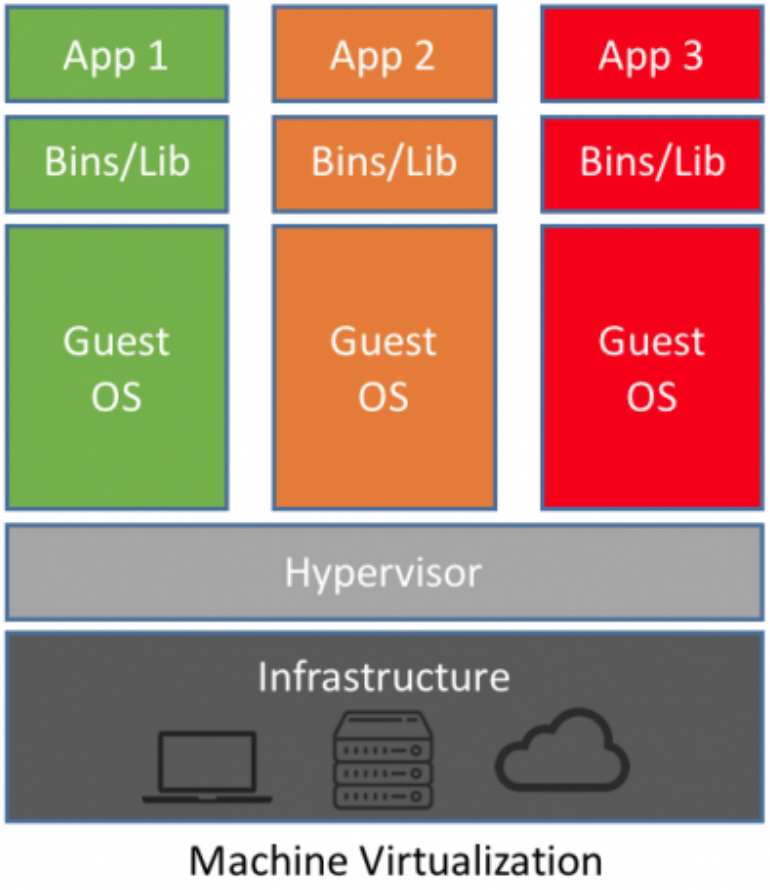
- Affirmation 2 :

> *Containers do not contain*
> *(Daniel Walsh)*

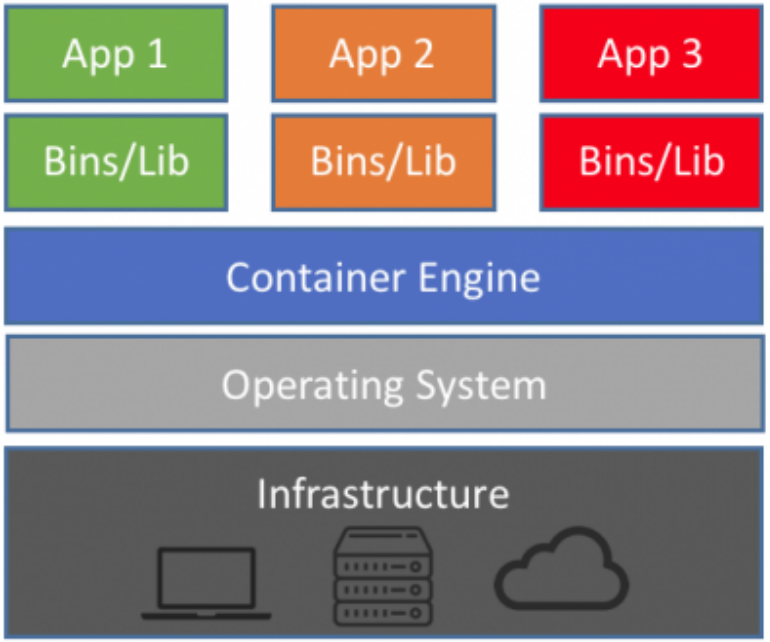
CONTENEURISATION VS VIRTUALISATION

- **Conteneurisation :** kernel de l'hôte partagé avec tous les conteneurs
- **Virtualisation :** kernel spécifique à chaque VM + matériel complètement émulé

CONTENEURISATION VS VIRTUALISATION



Machine Virtualization



Containers

INTÉRÊTS DES CONTENEURS

- Isolation des processus
- Reproductibilité des environnements
- Test et déploiement rapides

EXEMPLES DE TECHNOLOGIES DE CONTENEURISATION

- LXC
- Docker
- CoreOS Rkt

BRIQUES DE BASE DES CONTENEURS

- **Segmentation :** kernel namespaces
- **Limitation des ressources :** cgroups
- **Restrictions :** root capabilities, MAC, Seccomp, user namespaces…

NAMESPACES (1)

- **IPC (Inter Process Communications) :** concerne SystemV IPC et les files de messages POSIX
- **Network :** crée une nouvelle pile réseau virtuelle (IDs d'interfaces, règles de pare-feu, tables de routage...)
- **Mount :** isole les points de montage d'un processus

NAMESPACES (2)

- **PID :** chaque conteneur a son arbre (isolé) de processus. PID différents dans l'hôte et dans le namespace du conteneur.
- **User :** isole les UIDs et GIDs. `root` (UID 0) dans le conteneur n'est pas `root` en dehors.
- **UTS (UNIX Time Sharing) :** nom d'hôte et nom de domaine différents
- **Cgroup namespaces :** cache l'identité du cgroup dont fait partie un processus

CGROUPS (CONTROL GROUPS)

- *~ ulimits on steroids*
- Mécanisme hiérarchique et héritable
- Gère les ressources et les accès aux périphériques :
  - Mémoire
  - CPU
  - Disque
  - E/S de périphériques
  - Trafic réseau
  - ...

CAPABILITIES

- Rôle `root` divisé en plusieurs sous-sections de permissions
- Granularité plus fine des droits accordés pour les opérations privilégiées comme :
  - Créer une interface réseau
  - Monter un système de fichiers
  - Utiliser setuid
  - ...
- Un processus est lié à un ensemble de capabilities
- Processus enfants héritent de cet ensemble de capabilities

MAC (MANDATORY ACCESS CONTROLS)

- **Renforcement des contrôles de sécurité :** défense en profondeur et sécurité par niveau de permission dans le conteneur
- **LSM (Linux Security Modules) :** ajoute des hooks là où des appels système doivent accéder à un objet important du noyau
- **Exemples de LSM :** AppArmor (utilisé par défaut dans LXC et Docker), SELinux…

MAC : L'EXEMPLE D'APPARMOR

- **Profils AppArmor :** limitent voire bloquent :
  - les actions d'un programme sur les fichiers et les fonctions du système
  - le démarrage de processus (`pivot_root` et mount namespace)
  - le montage de certains types de FS
  - la modification dans certains emplacements comme `/sys` ou certaines parties de `/proc`
  - l'accès en lecture/écriture à la mémoire de l'hôte (`kcore`, `kmem`, `mem`...)
  - ...

SECCOMP

- Filtre les appels système
- Deux modes :
  1. *Strict Mode*: filtre = ensemble restreint d'appels système non customisables
  2. *Filter Mode* : filtre = programme écrit en BPF (Berkeley Packet Filter → Seccomp-BPF)
- Peut empêcher de manipuler le kernel (`init_module`...), de remplacer le kernel courant par une autre image kernel (`kexec_load`), d'utiliser du "kernel keyring" (`add_key`, keyctl`...)

DÉMOS

BONNES PRATIQUES DE SÉCURITÉ SUR DOCKER (1)

- Ne pas utiliser le flag `--privileged` mais plutôt :
  - `--cap-drop ALL --cap-add [...]`
  - `--security-opt apparmor=[...]`
  - `--security-opt=no-new-privileges`
- Désactiver la communication entre conteneurs :
  - `--icc=false`
- Lancer les conteneurs avec un FS en lecture seule :
  - `--read-only`
- Limiter les ressources :
  - `--ulimit nproc=<number>`

BONNES PRATIQUES DE SÉCURITÉ SUR DOCKER (2)

- Lancer les conteneurs en tant qu'utilisateur non root (directive USER) ou utiliser des user namespaces (`--userns-remap=default` dans le démon Docker)
- Ne pas monter des emplacements sensibles, surtout la socket du démon Docker...

BONNES PRATIQUES DE SÉCURITÉ SUR DOCKER (3)

- ⚠ Images préconstruites ⚠ :
  - Sources de confiance sur Docker Hub
  - Scan des images avec le Security Scanning de DTR
  - Image de base minimale (Alpine) 👌
- Mises à jour régulières
- Durcissement
- Minimisation de la surface d'attaque (pas de paquets inutiles par ex.)
- Tests automatisés (Docker Bench for Security)

FUTUR PROCHE : CONTENEURS *ROOTLESS* (1)

- **Conteneurs *rootless* :** Docker, Podman (Red Hat)…
- **Idée :**
  - Conteneurs non privilégiés…
  - Pouvant être créés et lancés par des utilisateurs non privilégiés…
  - Grâce à des *containers runtimes* et des orchestrateurs exécutés sans privilèges

FUTUR PROCHE : CONTENEURS *ROOTLESS* (2)

- **Objectifs :**
  - Plus de sécurité en cas de vulnérabilité des *runtimes* ou des orchestrateurs
  - Isoler des conteneurs imbriqués (*Docker-in-Docker*)
- Toujours plus ou moins au stade expérimental
- https://rootlesscontaine.rs/

QUESTIONS ?

SOURCES ET LECTURES RECOMMANDÉES

PRINCIPALES SOURCES

- NCC Group - Aaron Grattafiori : Understanding and Hardening Linux Containers
- NCC Group - Jesse Hertz : Abusing Privileged and Unprivileged Linux Containers
- MISC n°95 - Janvier / Février 2018 : *Docker : quelle sécurité pour les conteneurs ?*

SOURCES ET LECTURES RECOMMANDÉES

INTRODUCTION AUX NAMESPACES, CGROUPS ET CAPABILITIES (1)

- Jean-Tiare Le Bigot :
  - Introduction to Linux namespaces - Part 1: UTS
  - Introduction to Linux namespaces - Part 2: IPC
  - Introduction to Linux namespaces - Part 3: PID
  - Introduction to Linux namespaces - Part 4: NS (FS)
  - Introduction to Linux namespaces – Part 5: NET
  - Introduction to seccomp: BPF linux syscall filter

SOURCES ET LECTURES RECOMMANDÉES

INTRODUCTION AUX NAMESPACES, CGROUPS ET CAPABILITIES (2)

- Docker - Paul Novarese : Introduction to User Namespaces in Docker Engine
- Rami Rosen : Resource management: Linux kernel Namespaces and cgroups
- Nigel Brown : Using Linux Namespaces to Isolate Processes
- Red Hat - Joe Brockmeier : Secure Your Containers with this One Weird Trick

SOURCES ET LECTURES RECOMMANDÉES

INTRODUCTION AUX NAMESPACES, CGROUPS ET CAPABILITIES (3)

- Jérôme Petazzoni : Anatomy of a Container: Namespaces, cgroups & Some Filesystem Magic
- SCHUTZWERK - Philipp Schmied : An Introduction to Linux Containers

SOURCES ET LECTURES RECOMMANDÉES

POUR JOUER AVEC LES CONTENEURS, LES CAPABILITIES ET LES APPELS SYSTÈME

- Genuine Tools : amicontained (outil d'introspection de conteneur)
- Genuine Tools : contained.af ("jeu" de découverte des capabilities et des appels système)

SOURCES ET LECTURES RECOMMANDÉES

À PROPOS DE LA SÉCURITÉ (OU NON) DE DOCKER

- Daniel J Walsh : Are Docker containers really secure?
- Maciej Lasyk : "Containers do not contain"
- Jérôme Petazzoni :
  - Is it safe to run applications in Linux Containers?
  - Containers, Docker, and Security
  - Linux Containers (LXC), Docker, and Security
- Aaron Grattafiori : The Golden Ticket: Docker and High Security

SOURCES ET LECTURES RECOMMANDÉES

DOCKER ESCAPES (1)

- Trail of Bits : Understanding Docker container escapes
- Adam Zabrocki : Recent container escape and LKRG
- Dejan Zelic : The Danger of Exposing Docker.Sock
- Rory McCune : The Dangers of Docker.sock

SOURCES ET LECTURES RECOMMANDÉES

DOCKER ESCAPES (2)

- CyberArk - Nimrod Stoler : How I Hacked Play-with-Docker and Remotely Ran Code on the Host
- DragonSector - Adam Iwaniuk, Borys Popławski : CVE-2019-5736: Escape from Docker and Kubernetes containers to root on host
- Capsule8 - Brandon Edwards, Nick Freeman : A Compendium of Container Escapes

SOURCES ET LECTURES RECOMMANDÉES

CONTENEURS *ROOTLESS*

- Aleksa Sarai : Rootless Containers
- Akihiro Suda (NTT), Giuseppe Scrivano (Red Hat) : Rootless Containers