Audit de sécurité d'un environnement Docker

Julien RAEIS et Matthieu BUFFET

SSTIC 2018

14 juin 2018

Docker en 30 secondes

Système de « virtualisation légère »

Buts : paquetage d'applications, automatisation du cycle de développement, etc.

- réutilisation de briques applicatives (conteneurs) ;
- 1 brique applicative $\Leftrightarrow$ 1 conteneur $\Leftrightarrow$ 1 processus ;
- chaque conteneur embarque tous les fichiers dont il dépend ;
- conteneurs exécutés sur un même noyau mutualisé, assurant leur isolation ;
- leur vision est restreinte à un sous-ensemble de ressources du système.

Objectifs de la présentation

Cette présentation n'est pas...

- une introduction à l'utilisation de Docker ;
- une revue détaillée des mécanismes bas niveau sous Linux ;
- un nouveau guide de configuration ou de durcissement de Docker :
  - la littérature en regorge (cf. *CIS Benchmarks*),
  - de même que des outils pour la conformité (cf. `docker-bench-security`[a]) ;
- une présentation d'un outil d'audit clés en main.

a. https://github.com/docker/docker-bench-security

Objectifs de la présentation

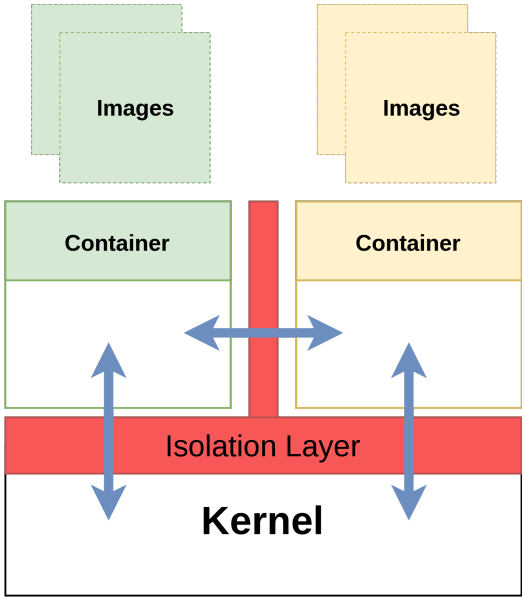
On va parler...

- de Docker sous Linux **et** Windows, comme cas concret d'application ;
- de méthodologie d'audit :
  - quel points auditer ?
  - vis-à-vis de quelles menaces ?

Constats

- images pas toujours maîtrisées côté développeur ;
- durcissement par défaut générique : satisfaisant, mais pas suffisant ;
- guides de durcissement actuels comblent certaines lacunes, mais pas toutes ;

↪ objectif : durcir le paramétrage d'exécution des conteneurs



Linux

Mécanismes d'isolation

cgroups

- restriction des ressources qu'un processus peut consommer ;
- notamment `devcg` pour les périphériques.

namespaces

- restriction de ce qu'un processus voit du système ;
- en particulier PID, NET, IPC, MOUNT et USER *namespaces*.

Mécanismes de sécurité

Principe du moindre privilège traduit au travers :

- des *capabilities* : découpage des droits `root` ;
- de *seccomp* : filtrage des appels systèmes autorisés ;
- MAC comme SELinux ou AppArmor : politique de sécurité globale au système.

Windows

Historique

Travail conjoint de Microsoft et Docker Inc. depuis fin 2014

- au début, exécution de conteneurs Linux dans Hyper-V ;
- conteneurs Windows natifs depuis Windows Server 2016 Technical Preview 5.

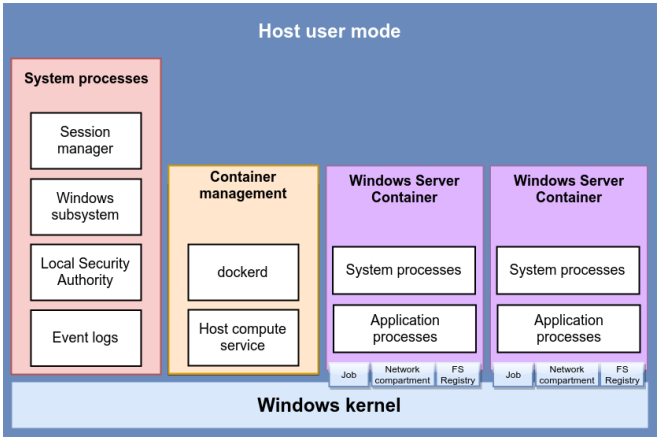
Deux types de conteneurs :

- Windows Server Containers (WSC) sous Windows Server 2016 uniquement ;
- Hyper-V Containers sous Windows Server 2016 et depuis Windows 10 1607.

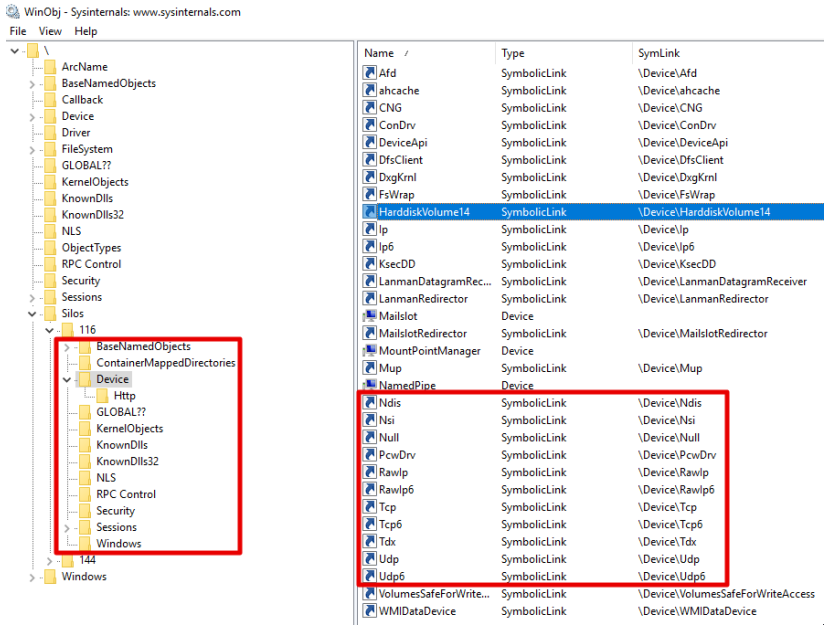
Windows Server Containers

Conteneurs basés sur des objets de type `Job`

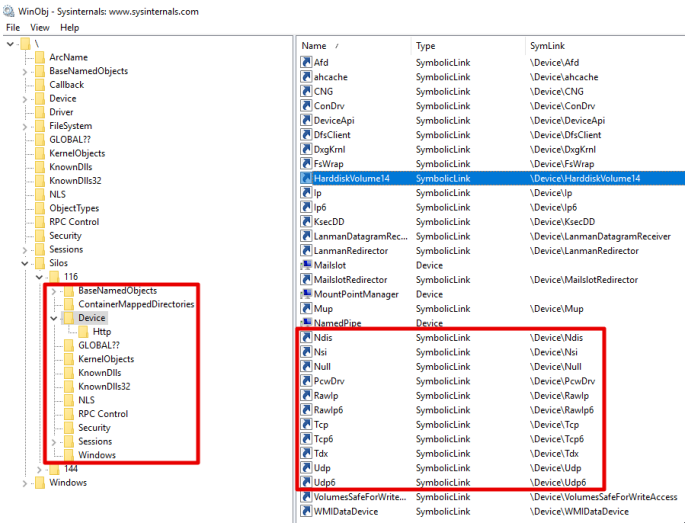
- structure étendue avec la notion de *silo* ;
- possède une *session* dédiée ;
- permet la virtualisation de l'espace des *objets noyau* ;
- certains pilotes et portions du noyau sont modifiés pour vérifier l'état de silo ;
- interfaces réseau et règles pare-feu isolées dans des *Network Compartments* ;
- dans Docker, `--isolation=process` (cas par défaut sous Windows Server !).



```
docker run -it --isolation=process microsoft/windowsservercore
```



```
docker run -it --isolation=process microsoft/windowsservercore
```

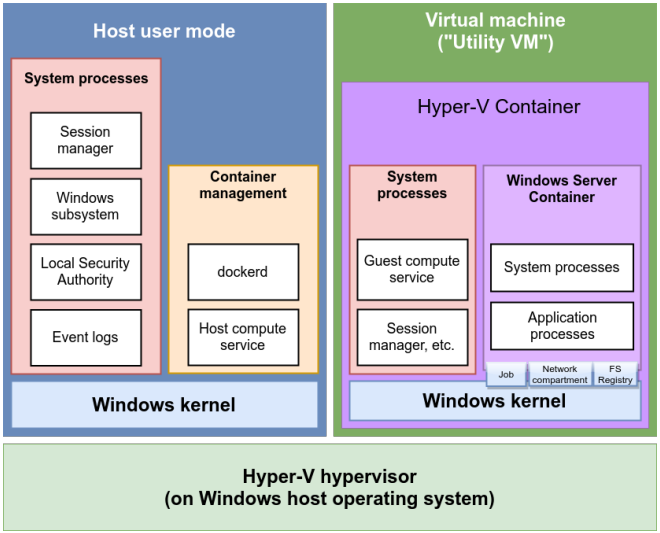



```
kd> !silo
Address           Type       ProcessCount Id
ffff8f820b040080 ServerSilo 20     [...] (0n116)
ffff8f820b792810 ServerSilo 17     [...] (0n144)
2 active Silo(s)

kd> !object \Silos\116\Device\Tcp
Type: (ffff8202412633a0) SymbolicLink
[...]
Flags: 0x000001 ( Global )
Target String is '\Device\Tcp'

kd> !object \Silos\116\Device\NamedPipe
Type: (ffff820241346f20) Device
[...]
```

Hyper-V Containers

Exécution d'un Windows Server Container dans une machine virtuelle Hyper-V :

- une instance du noyau Windows Server par conteneur ;
- mécanismes WSC + isolation d'Hyper-V ;
- interfaces Ethernet virtuelles d'Hyper-V ;
- communication avec l'hôte (volumes, etc.) par VMBus ;
- dans Docker, `--isolation=hyperv`.



Interfaces de contrôle Docker

Menaces et risques

- accès non-autorisé à l'une des interfaces de contrôle (socket UNIX ou TCP/IP) ;
- instanciation de conteneurs, élévation de privilèges, évasion.

Remédiation

- groupe `docker` (ou `docker-users`) minimisé, permissions restreintes sur socket UNIX ;
- authentification mutuelle TLS obligatoire sur socket TCP/IP ;
- mise en place d'autorisations par plugin (ex. *Twistlock*).

Points d'audit

- sockets en écoute
- moyens d'authentification
- éventuelle politique d'autorisations sur l'API

Interfaces de contrôle Docker - démo

Élévation locale de privilèges

- hypothèse : l'attaquant a un compte sur l'hôte dans le groupe `docker`/`docker-users` ou peut accéder à la socket ;
- exemple sous Linux : https://fosterelli.co/privilege-escalation-via-docker.html ;
- démonstration sous Windows.

Ressources système

Menaces et risques

- déni de service sur un hôte, ses conteneurs, ou les nœuds d'un *swarm* ;
- accès illégitime à des périphériques du système hôte.

Remédiation

- mise en place de limitations de consommation des ressources (CPU, mémoire, etc.) : *cgroups* / *ulimit* sous Linux, Jobs sous Windows ;
- contrôle d'accès aux périphérique au travers de `devcg` (lecture / écriture / création).

Points d'audit

- limites mises en œuvre pour les conteneurs (*docker inspect*, *Procexp*) ;
- auditer l'ensemble des périphériques partagés (`/sys/fs/cgroup/devices/devices.list`).

Isolation fournie par les *namespaces*

Menaces et risques

- accès à des ressources de l'hôte (réseau, processus, système de fichiers) ;
- élévation de privilèges.

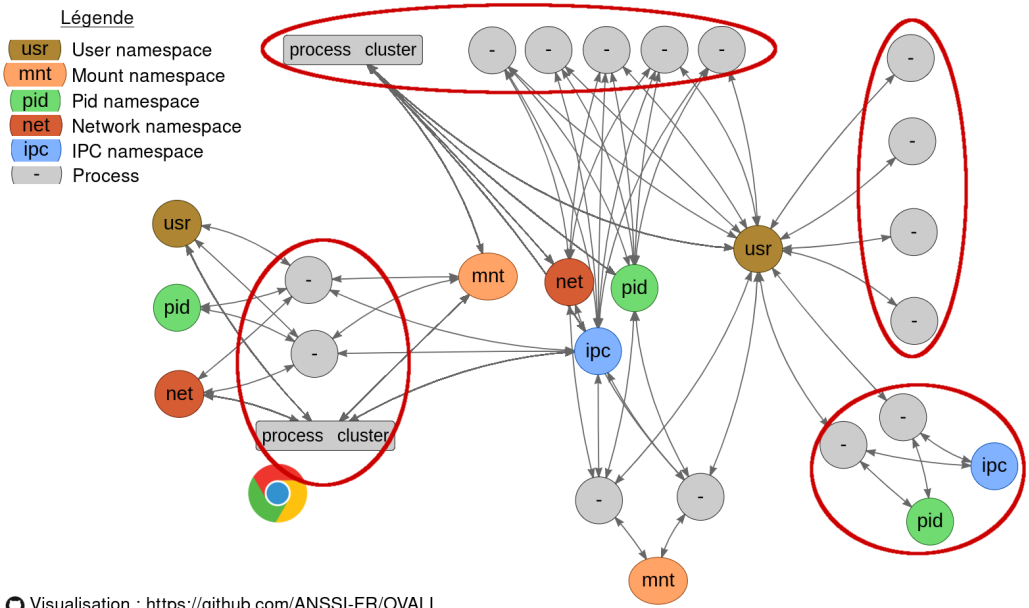
Remédiation

- PID namespace (ex. ptrace) ;
- IPC namespace (ex. shared memory segments) ;
- Mount namespace (ex. /proc) ;
- Net namespace (ex. interception de trafic) ;
- User namespace (ex. mauvaises configurations des capabilities, des volumes).

Points d'audit

- membres de chaque *namespace* ;
- qualification des éventuels partages :
  - de *namespaces* avec l'hôte ou d'autres conteneurs (`--pid=container:<id>`),
  - de ressources de l'hôte : socket Docker, socket X11.

Isolation fournie par les *namespaces* - outils



Visualisation : https://github.com/ANSSI-FR/OVALI

Liste des *capabilities* (Linux)

Menaces et risques

Évasion et élévation de privilèges

Remédiation

- principe du moindre privilège ;
- liste restreinte par défaut dans Docker, mais profil reste trop générique ;

Points d'audit

Liste des *capabilities* :

- effectives pour les processus en exécution d'un conteneur (`pscap` / `capsh`) ;
- ajoutées/supprimées par rapport à la liste par défaut de Docker ;
- réellement nécessaires...

Liste des *capabilities* - outils

```
$ capable[1] &      (ou captrace[2])
$ docker run -it nginx
[...]
01:55:37  0      6628   docker-runc-cur  19   CAP_SYS_PTRACE          1
01:55:37  0      6628   docker-runc-cur  19   CAP_SYS_PTRACE          1
01:55:37  0      20738  dockerd-current  29   CAP_AUDIT_WRITE         1
01:55:37  0      6603   nginx             0   CAP_CHOWN               1
01:55:37  0      6603   nginx             0   CAP_CHOWN               1
01:55:37  0      6603   nginx             0   CAP_CHOWN               1
01:55:37  0      6603   nginx             0   CAP_CHOWN               1
01:55:37  0      6603   nginx             0   CAP_CHOWN               1
01:55:37  0      6603   nginx            10   CAP_NET_BIND_SERVICE    1
01:55:37  0      6634   nginx             6   CAP_SETGID              1
01:55:37  0      6634   nginx             6   CAP_SETGID              1
01:55:37  0      6634   nginx             7   CAP_SETUID              1
01:55:38  0      1476   Xorg             15   CAP_IPC_OWNER           1
01:55:38  0      1476   Xorg             15   CAP_IPC_OWNER           1

$ docker run --cap-drop=all --cap-add=chown,net_bind_service,setgid,setuid nginx
```

1. https://github.com/iovisor/bcc/blob/master/tools/capable.py
2. https://github.com/mtth-bfft/captrace

Ressources non abstraites par le conteneur

Menaces et risques

- Accès à des ressources non couvertes ou protégées par :
  - définition (ex. modules noyau, périphériques),
  - difficulté (ex. date-heure, *keyrings*),
  - erreur ☹.
- Évasion vers l'hôte

Remédiation

Réduction de la surface d'attaque de l'hôte

Points d'audit

Utilisation d'un mécanisme adapté de réduction de surface d'attaque de l'hôte :

- Hyper-V Containers et *mitigation policies* sous Windows (*Procexp*, *tristitude*[a]) ;
- application du couple *seccomp* et MAC sous Linux.

a. https://github.com/mtth-bfft/tristitude

Containers do not contain

Deux exemples d'évasion des Windows Server Containers

- par `NtGetNextProcess()`
  © A. Ionescu, SyScan 360, Mai 2017
  (patché dans Server 1709 et 1803... backport Windows Server 2016 ?)
- par `OpenFileById()`
  (patché dans Server 1803... backport Server 2016 et 1709 ?)

Attention, pour Microsoft les WSC ne sont pas une frontière de sécurité !

(2014 - Daniel Walsh)

Politique *seccomp* (Linux)

Menace et risques

- exploitation de vulnérabilités noyau ;
- évasion vers l'hôte.

Remédiation

- filtrage des appels systèmes par *seccomp* ;
- emploi d'un MAC pour atténuer les conséquences.

Points d'audit

- présence d'un profil *seccomp* activé ;
- désassemblage du filtre *seccomp* (attention : `--cap-add, --privileged` → *seccomp*) ;
- vérification du taux de couverture du filtre.

Politique *seccomp* - outils

Outils d'analyse de filtre *seccomp-bpf*

Affinement du filtre seccomp : trace d'appels système avec `strace -qcf` :

```
# strace -qcf /usr/sbin/apachectl -d . -f apache2.conf -DFOREGROUND
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
38.21    0.030050           7      4519         4 openat
24.33    0.019129           4      4576           close
23.99    0.018864           4      4558           read
 3.28    0.002578           8       311           munmap
 3.02    0.002377           5       453         1 mmap
 0.43    0.000338           5        69        69 access
 0.35    0.000272           8        35         8 connect
 0.27    0.000210           6        33           socket
[...]
 0.00    0.000000           0         1           listen
 0.00    0.000000           0         3           execve
------ ----------- ----------- --------- --------- ----------------
100.00    0.078637                 16097       128 total
```

Politique *seccomp* - outils

Outils d'analyse de filtre *seccomp-bpf*

Prototype d'analyseur formel en Prolog [a] (Linux 4.4+) : exemple sur le filtre Docker par défaut

a. https://github.com/mtth-bfft/seccomp-analyze

```
# seccomp-dump -f prolog -p $PID >filter.pl
# swipl ./seccomp.pl
?- [filter].
?- filter_accepts(175, 3221225534, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6).
false.                    ("can I call init_module() on x86_64 with some arguments?")
?- filter_accepts(Nr, Arch, Rip, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6), [...]
Nr=66 Arch=1073741827
Nr=70 Arch=1073741827
Nr=71 Arch=1073741827
Nr=75 Arch=1073741827
Nr=76 Arch=1073741827
Nr=77 Arch=1073741827
[...]
```

Récapitulatif des points d'audit

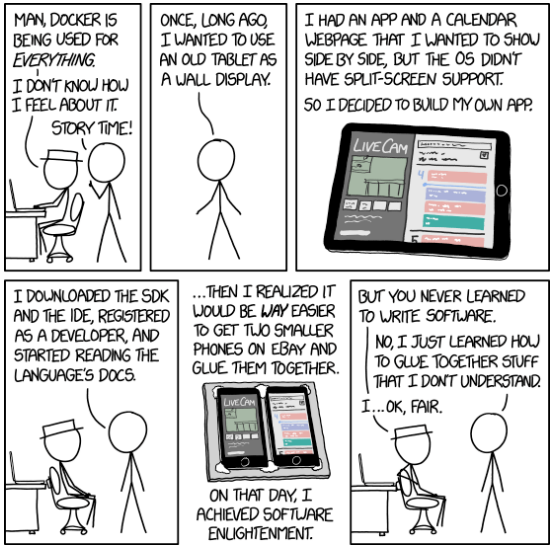
	Déni de service	Évasion	Ressources non autorisées	
Interfaces de contrôle		●	●	Authentification mutuelle Autorisations sur les interfaces
Ressources système (consommation)	●			Limites d'utilisation des ressources
Ressources système (périphériques)	●	●	●	Contrôle d'accès aux périphériques
Isolation des processus		●	●	Activation des *namespaces* Partage éventuel de *namespaces*
Privilèges des processus	●	●	●	Limitation des *capabilities*
Ressources non abstraites		●	●	Utilisation des *Hyper-V Containers* Mise en place d'un MAC
Vulnérabilités noyau	●	●	●	Application d'un filtre *seccomp* Mise en place d'un MAC Mise à jour du système hôte

Pour aller plus loin

Autres pistes à creuser

- Orchestrateurs (Docker Swarm, Kubernetes) ;
- Katacontainers (ex-Intel ClearContainers) : équivalent Linux aux Hyper-V containers ;
- gVisor : noyau en mode utilisateur de Google à destination des conteneurs ;
- WDAG : bac à sable de Microsoft Edge basé sur les Hyper-V Containers ;
- Cilium : filtrage réseau inter-conteneurs au travers de filtres eBPF ;
- SGX : protection des conteneurs vis-à-vis de l'hôte (cf. projets *Haven*, *SCONE* : sévères limitations).

Questions ?



Source : XKCD - CC BY-NC 2.5 (https://xkcd.com/1988)