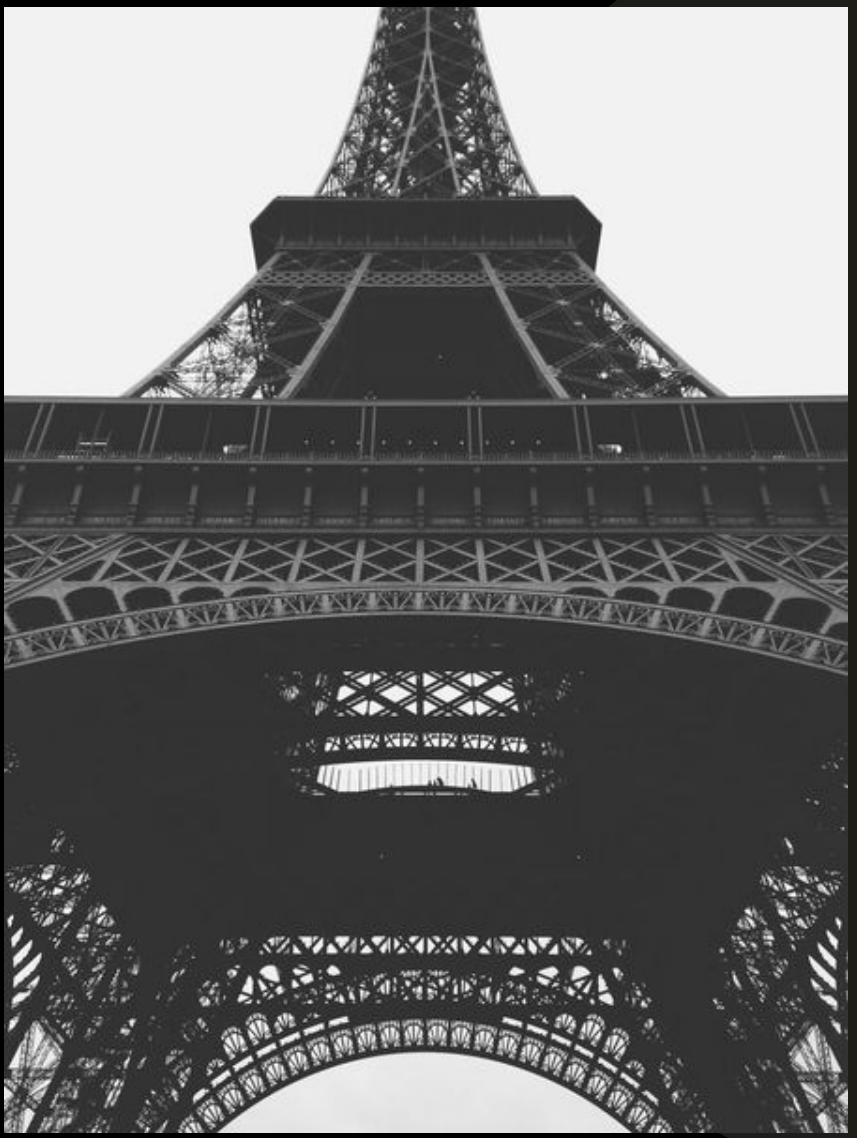
Meetup SecOps#1

Lundi 17 juin 2019

DURCISSEMENT* DES CONTENEURS

UNE ENTRÉE EN MATIÈRE

* HARDENING



DURCISSEMENT DES CONTENEURS

1. Les 7 propriétés d'un système robuste
2. Historique, avantages et maturité
3. Vulnérabilités

4. Abstraction OS: Virtualisation à Micro-Segmentation

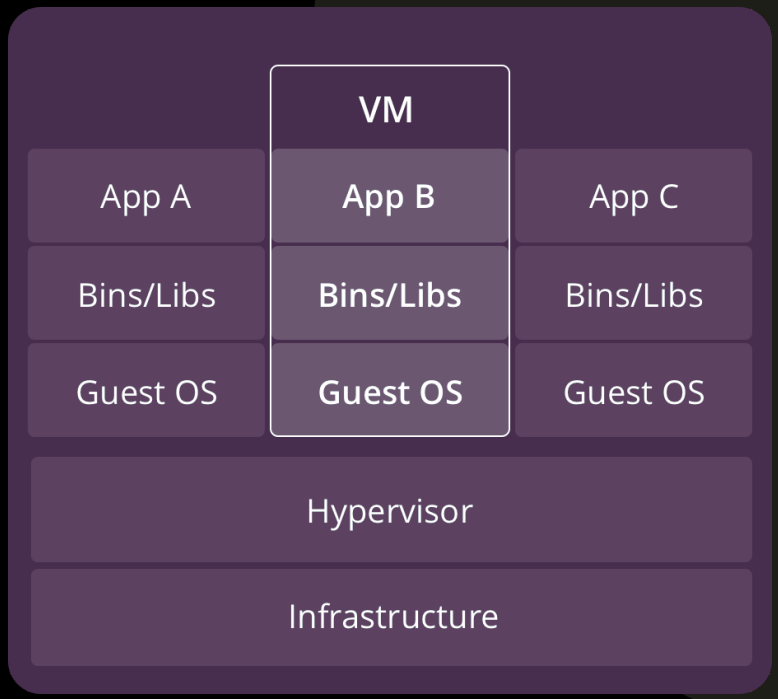
5. Les « Must-have » pour déployer du conteneur
6. Les bonnes pratiques
7. Solutions de type « sidecars »
8. Solutions de type runtime

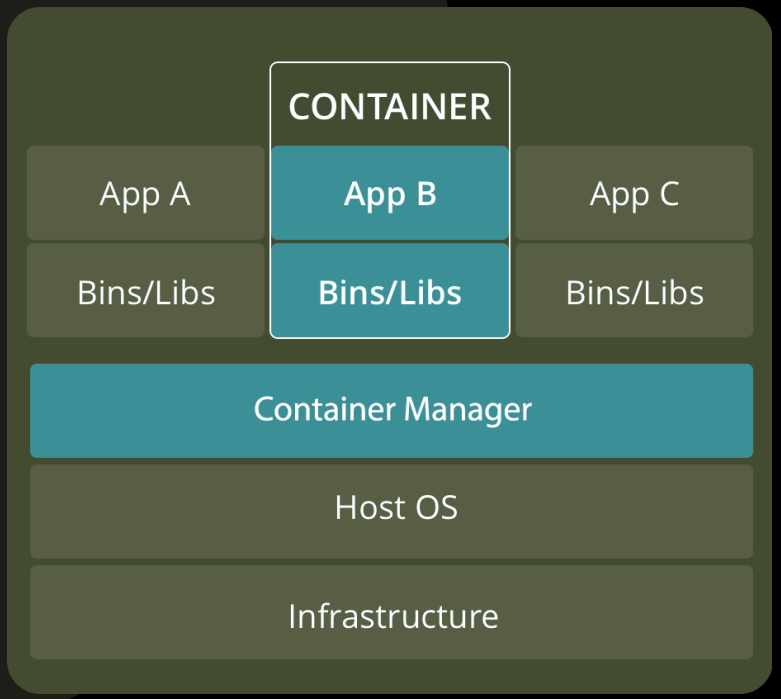
LES 7 PROPRIÉTÉS D'UN SYSTÈME ROBUSTE

 Hardware-based Root of Trust / RoT	Clé de chiffrement racine inaliénable (UEFI BIOS) Protège matériellement l'intégrité système => Confiance dans l'OS (pas de rootkits)
 Trusted Computing Base / TCB	Logiciels/utilitaires utilisant des élévations de privilèges (clés privés dans un Vault) – 'orange box'
 Défense en profondeur	Plusieurs mesures possibles, en cas de faille d'une des défenses, limitera l'impact de l'attaque
 Compartimentation	Barrières matérielles entre les éléments logiciels afin d'éviter une propagation
 Authentification basée sur un certificat-client	Certificat de type SSH prouvant l'identité et l'authenticité
 Système de sécurisation adaptable	Permet de renouveler, mettre à jour de manière sûr, les parties de logiciels, compromises par des vulnaribilités
 Système d'alertes des erreurs système/logiciel	Repérer les attaques , ou les tests sur des failles (buffer overrun)

CONTENEUR : POURQUOI ?

Image d'un conteneur ne contient pas l'OS (isolation soft par namespace)
=>Léger et performance native (si sur baremetal)
⇒ Vélocité: millisecondes pour un docker start / un docker stop
⇒ Construction d'une image avec un fichier Dockerfile (en + : Infrastructure As Code)
⇒ Elasticité possible avec Kubernetes => **COUTS optimisés**





MATURITÉ DES CONTENEURS (ET DE SON ÉCOSYSTÈME)

- 2007 Dotcloud (PaaS)
- 2008 1er version de LXC / cgroups
- 2013 Docker open source
- 2014 Docker 1.0
- 2018: choix de Kubernetes par les clouds providers(AWS, Azure, GCP, OVH, …), RedHat,Pivotal, VMWare, IBM
  - 31,000 développeurs
- 03/2019: Docker 18.09.4 / 1.7Millions de développeurs actifs par mois

- Le futur:
  - Cloud Foundry pour les microservices/serverless
  - CF n'expose pas les conteneurs comme le fait K8s
  - K8s supportant des micro-VM

HISTORIQUE DOCKER

- Docker 0.5.2 : utilisation du root pour l'utilisation de socket
- Ouverture du docker registry, peu de contrôles des images / non signées

⇒Mauvaises réputations

⇒réactions:

- Docker Trusted registry / Docker Hub
- Possibilité de définir des user namespaces UID => 1er niveau d'isolation

EXEMPLE DE VULNÉRABILITÉS

- CVE-2019-5736

Impact:	Important
Public Date:	2019-02-11
CWE:	CWE-672

Bugzilla:

1664908: CVE-2019-5736 runc: Execution of malicious containers allows for container escape and access to host filesystem

Sur Docker 18.09.2, LXC, Apache Mesos, cri-o et containerd

EXEMPLE DE VULNÉRABILITÉS

- alpine CVE-2019-5021

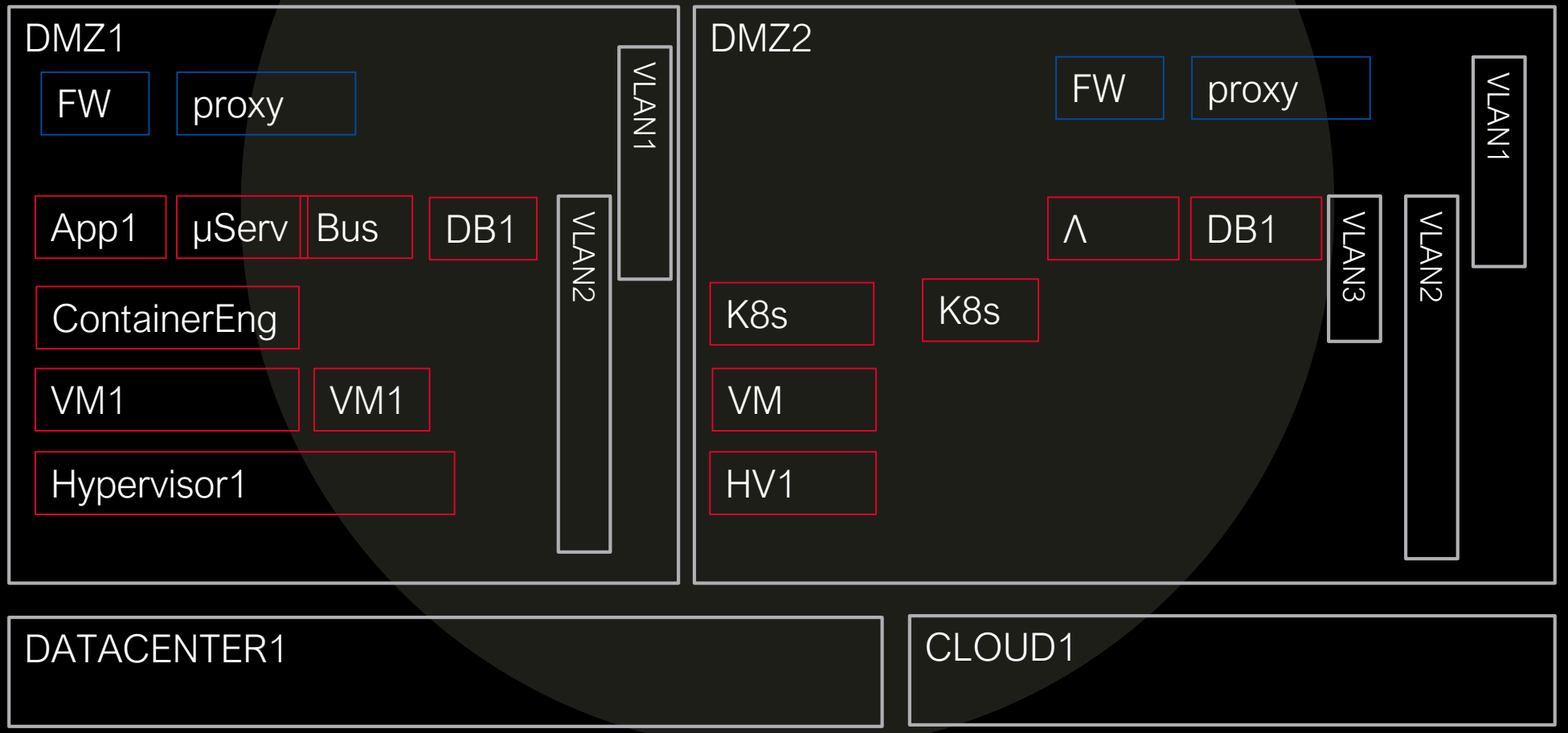
CVE-ID	
CVE-2019-5021	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
Versions of the Official Alpine Linux Docker images (since v3.3) contain a NULL password for the `root` user. This vulnerability appears to be the result of a regression introduced in December of 2015. Due to the nature of this issue, systems deployed using affected versions of the Alpine Linux container which utilize Linux PAM, or some other mechanism which uses the system shadow file as an authentication database, may accept a NULL password for the `root` user.	

EXEMPLE DE VULNÉRABILITÉS

- Différents types :
  - Kernel exploits (=> kernel panic sur toute la plateforme!)
  - Denial of Service
  - Casse du container (surtout si le container a les privilèges root / pas de namespace UID par défaut)
    - Et Prise de contrôle ,,,
  - Images corrompues
  - Secrets/mots de passe exposés

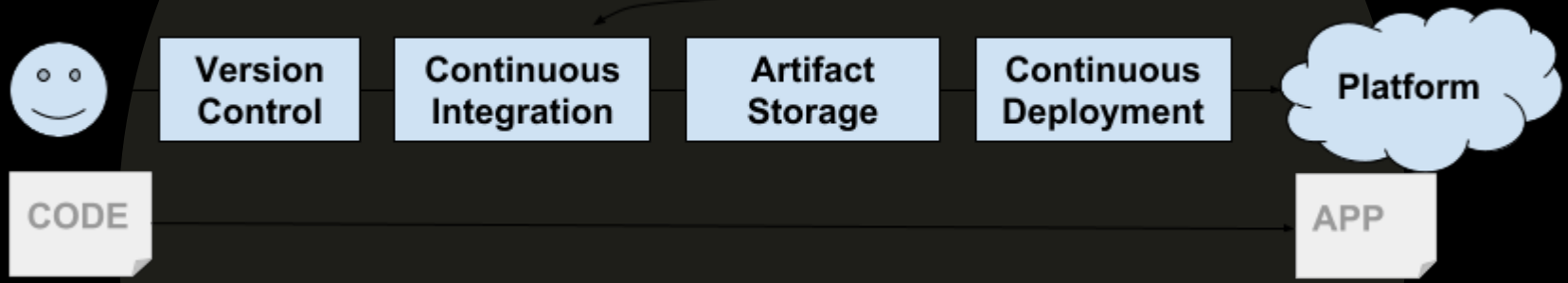
MICRO-SEGMENTATION

- Utiliser une VM et non un serveur physique comme hôte d'un moteur de conteneurs
- Prérequis: Virtualiser le réseau / VLAN en statique(complexe) -> en dynamique (SD-WAN)



MUST-HAVE AVANT LES CONTENEURS!

- Une pipeline CI/CD pour *contrôler* la qualité et la traçabilité des livrables!



- Une supervision / suivi des logs systèmes et logiciels (bus de données ou moteur de recherche)
  - Microservices/ serverless => des tonnes des fichiers .log

BONNES PRATIQUES POUR LA SÉCURITÉ

- Parquer les conteneurs d'un même UID/user sur le même hôte
- Image Distroless (pas d'OS , que du déploiement applicatif) => surface d'attaque, vulnérabilités systèmes , secret sécurisé
- registry d'image privatif
  - comme docker trusted registry/ sonatype nexus/ gtilab/github/azure/amazon eleastic/google container registry
- Pas de mot de passe hardcodé / utiliser –secret
- Vérifier la signature des images (mais cela ne suffit pas)

```
gpg: Signature made Wed 15 May 2019 03:05:06 PM CEST
gpg:                using RSA key D75899B9A724937A
gpg: Good signature from "Nextcloud Security <security@nextcloud.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 2880 6A87 8AE4 23A2 8372  792E D758 99B9 A724 937A
```

- Utiliser docker inspect et Dive

BONNES PRATIQUES SUR DOCKERD

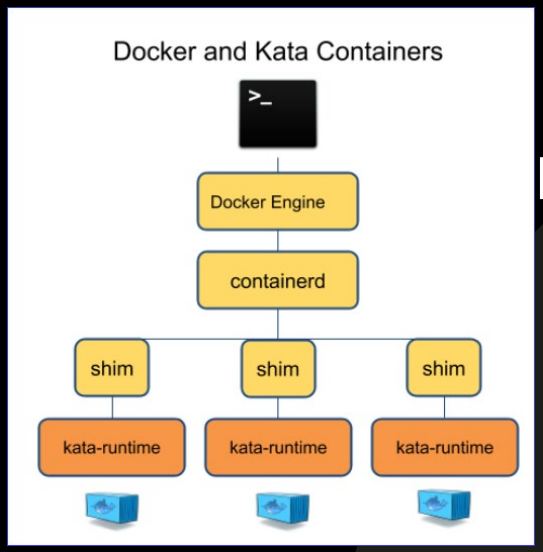
- Configurer les control groups (limiter les ressources par container => contre les DoS)
- Limiter l'accès au démon docker (moteur conteneur), supprimer les accès REST API, activer TLS sur le port de service (2375 par défaut)
  - Vérifier sa whitelist pour limiter les droits
  - N'activer les accès SSH ou cron que si nécessaire (car process en root)
  - Limiter / interdire les points de montage externe, les sockets niveau 2 ou 3
- Ajouter AppArmor , SELinux, GRSEC sur la distrib du dockerd
- https://docs.docker.com/engine/security/security

LES SOLUTIONS « SIDECAR »
- BUILD/~~RUN~~/MONITOR -

- Scan d'images dans le pipeline CI/CD
- Assurance d'avoir Immutabilité des fichiers du runtime (limiter les droits des conteneurs)
- Surveillance des activités suspectes
- Appliquer des règles de sécurité sur l'ensemble des conteneurs déployés

LES SOLUTIONS « SIDECAR »
LISTE PRODUITS

- AquaSec (agent en multitenant)
- Neuvector (détection paquet niveau 7)
- Sysdig (monitor et secure pour scan de vulnaribilités)
- Twistlock (defend / Monitor/manage => agent et dashboard compliance)
- StackRox (dashboard compliance)



LES RUNTIME/RUNC ALTERNATIFS COMPATIBLE OCI RUNTIME SPEC

Kata-container basé sur Intel Clear Containers

=> kata-runtime à la place de runc, et

un agent communiquant en gRPC

- gVisor (google)
- Nvidia/RedHat cri-o (GPU)
- Containerd / cri
- KubeVirt (exécuter des VMs sur k8s)
- Hybervisor based container frameworks
  - Hyper Container https://hypercontainer.io/
  - Clear Container https://github.com/clearcontainers/runtime/wiki
  - frakti https://github.com/kubernetes/frakti

LOG MANAGEMENT

- Prometheus / grafana
- ELK / elastic
- SumoLogic
- GuardDuty
- Datadog

INGRESS CONTROL SERVICE

- NGINX Ingress
- ISTIO envoy
- GLOO (https://gloo.solo.io/)
- TIGERA firewall (https://www.tigera.io/zero-trust/)
- F5
- …
- K8s Ingress service

THE LINUX FOUNDATION PROJECTS

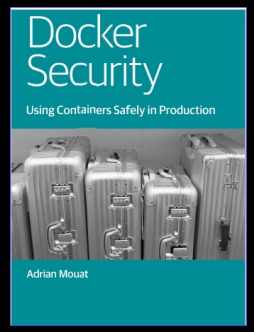
OPEN CONTAINER INITIATIVE

kubernetes

RÉFÉRENCES STANDARDS

- OCI Runtime Specification http://www.opencontainers.org
- Kubernetes CRI (Container Runtime Interface)
- https://kubernetes.io/blog/2016/12/container-runtime-interface-cri-in-kubernetes/

- Pattern Sidecar:
- https://docs.microsoft.com/en-us/azure/architecture/patterns/sidecar



RÉFÉRENCES

- Dockerfile best practices

https://github.com/GoogleContainerTools/distroless ou http://grc.io

https://github.com/wagoodman/dive

https://www.docker.com/dockercon/2019-videos?watch=dockerfile-best-practices

https://www.oreilly.com/ideas/docker-security

https://cloud.google.com/container-registry/docs/get-image-vulnerabilities

https://github.com/kata-containers

https://cri-o.io/

https://developer.nvidia.com/nvidia-container-runtime

https://github.com/containerd/cri