Hardening K8s

1.15+

19-02-2019

V2.1

Trusted partner for your **Digital Journey**

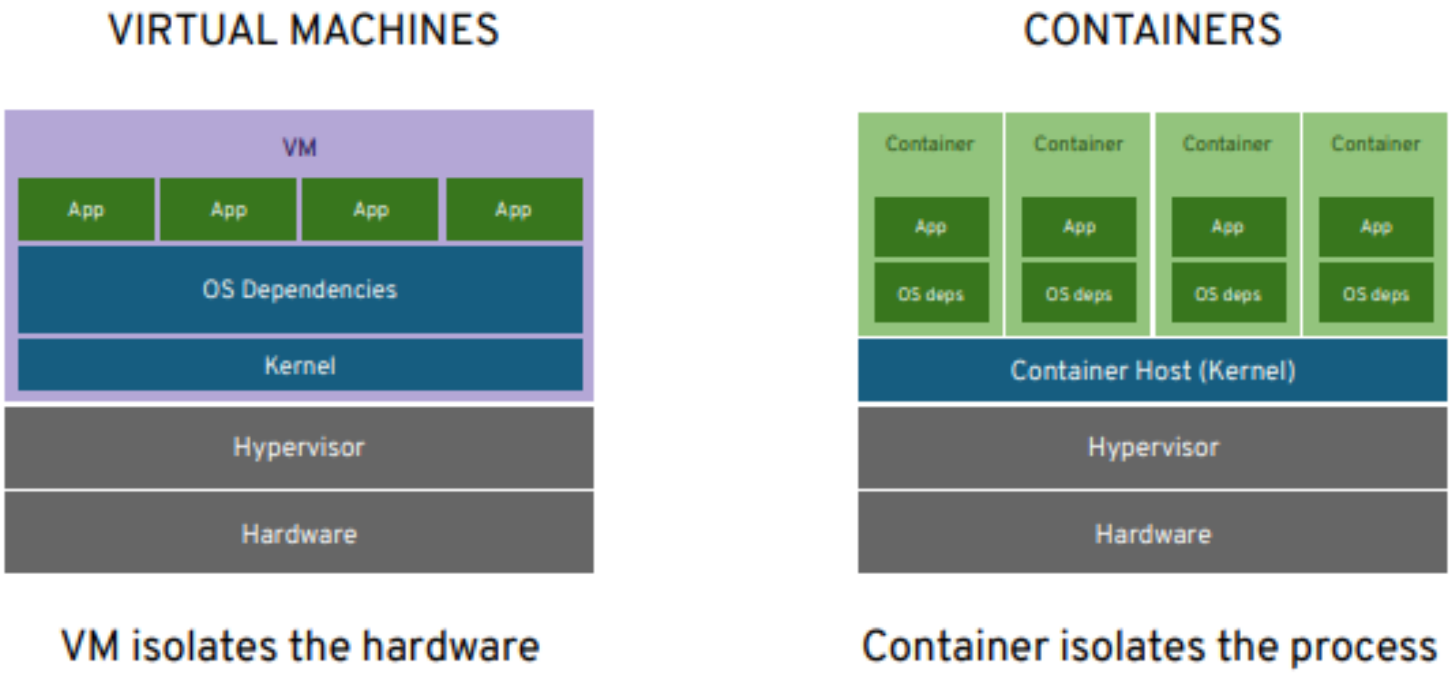
Atos

sommaire

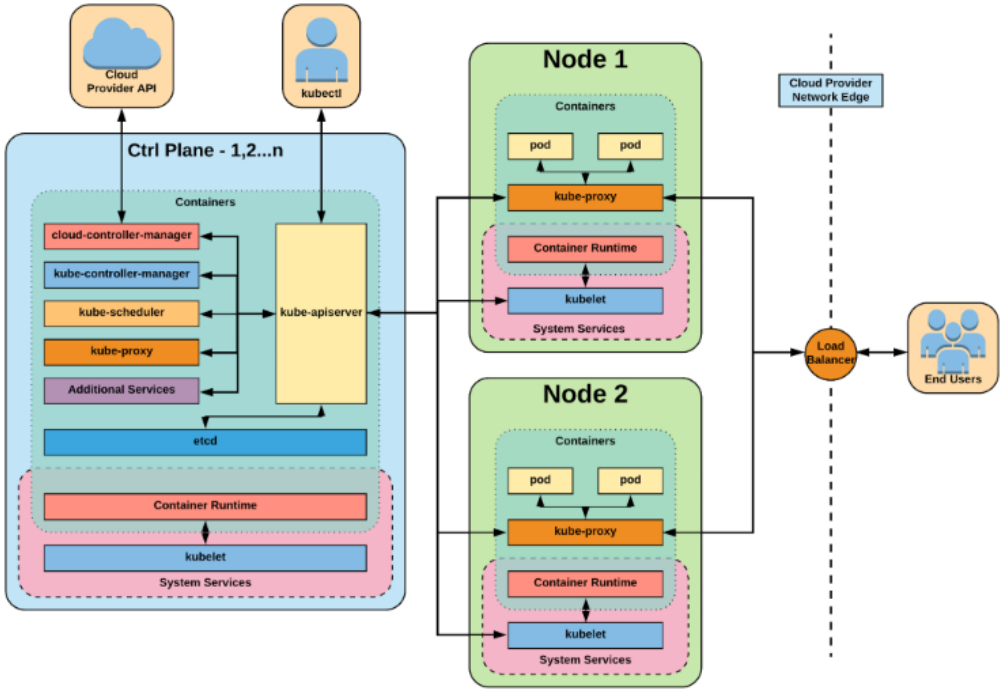
- Introduction
- Sécuriser la machine Host
- Sécuriser le Traffic
- Sécuriser le Control-Plane
- Sécuriser les composants internes
- Monitoring & Logging
- Perspectives
- Appendix

Introduction

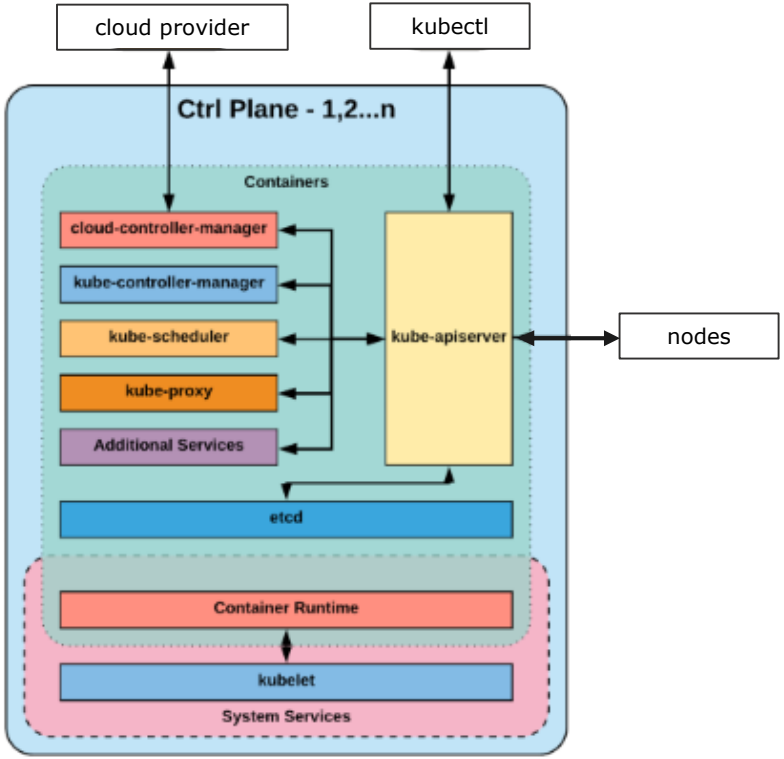
VMs & Containers



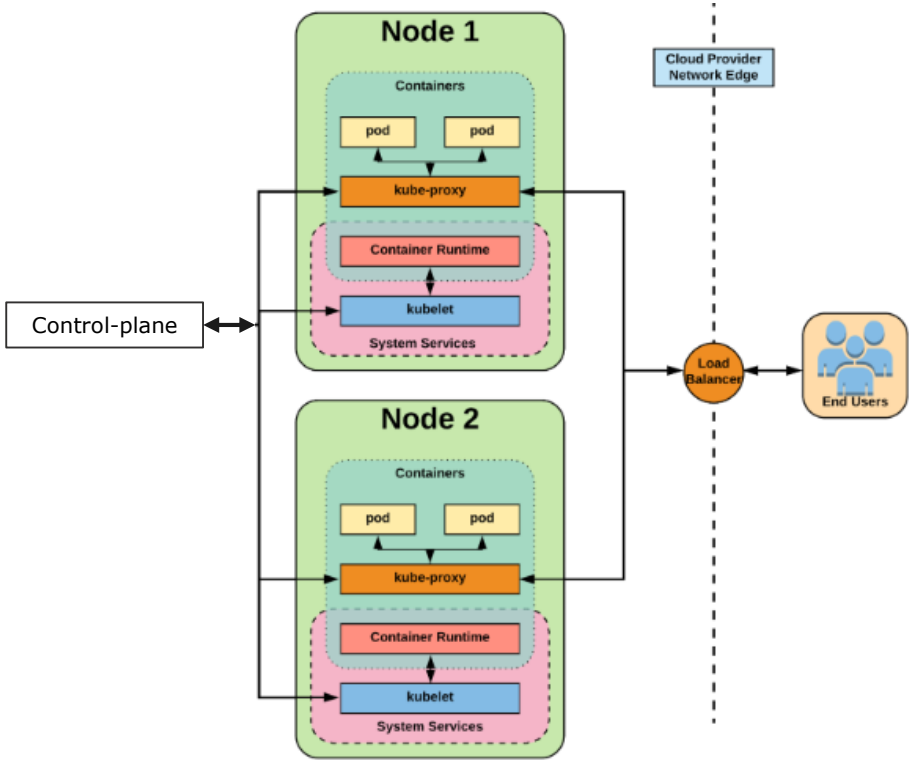
Architecture K8s



Architecture K8s : Control Plane



Architecture K8s : Components



Les risques ?

- Perdre le contrôle total du cluster
- Perte de données
- Usurpation d’identité
- Accès privilégié sur les hosts (déstruction du système)
- Accès non autorisé à des projets au sein du cluster
- Cryptocurrency mining

Sécuriser la machine Host

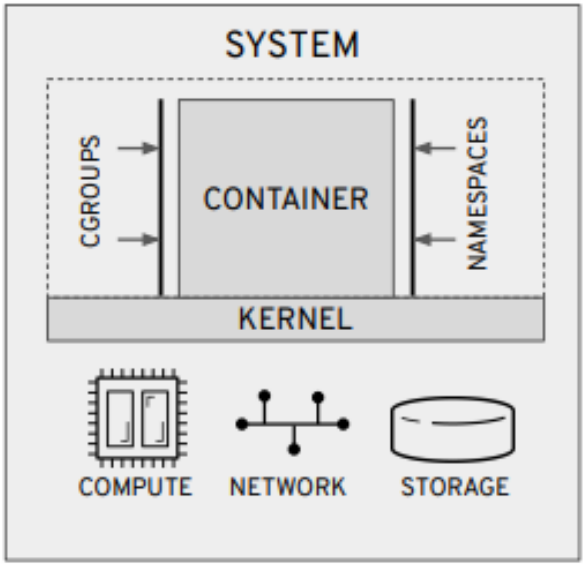
Sécurisation de la machine Host

Risques

- Accès aux configurations (kubelets, scheduler ..)
- Passer des requêtes non autorisés au serveur API
- Accéder au données d'autres conteneurs
- Détruire la totalité du système

Remédiation

- Montage des dossiers systèmes (/sys, /proc/sys,…) en Lecture-Seule
- Renforcer l'utilisation des cgroups (limiter les ressources systèmes, temps CPU …)
- Mise en place contexte de sécurité SElinux
- Limiter les appels systèmes avec Seccomp

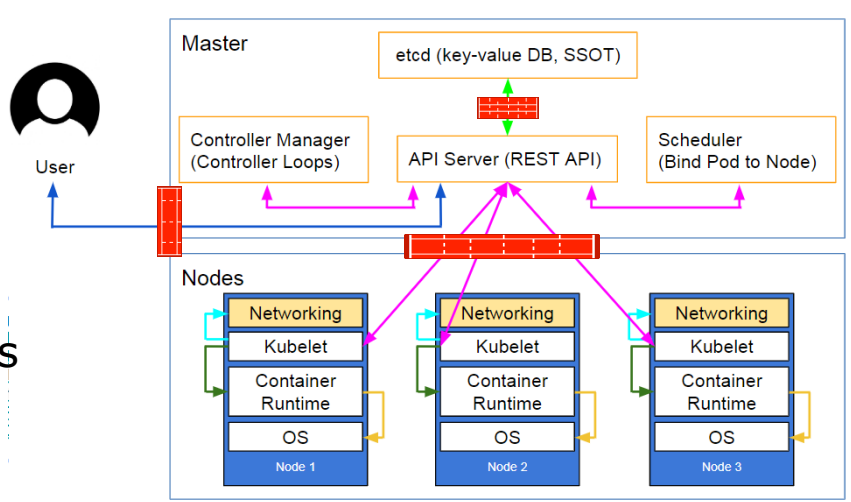


Sécuriser le Traffic

Sécuriser le Traffic: Transport Security

Risques

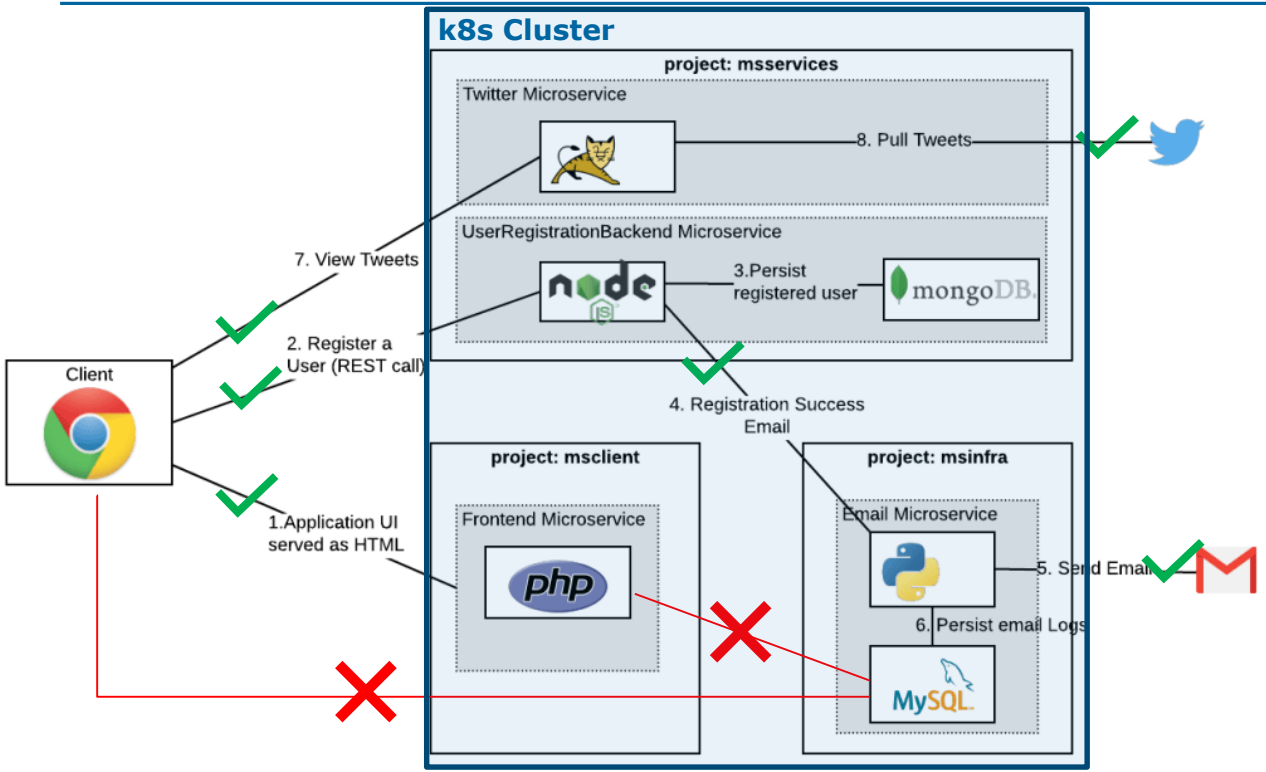
- Sniffing de données
- Vols d'identités / request forgery
- Connexions non autorisés entre conteneurs



Remédiation

- Toutes les communications doivent être protégées par des certificats TLS mutualisés
- Séparer le cluster kubernetes du cluster etcd et appliquer des règles de firewall
- Firewall interne et externe pour limiter les requêtes au serveur d'api
- Appliquer des politiques de networking (**NetworkPolicy**)

Sécuriser le Traffic: NetworkPolicy



Exemples de policies :

- refuser tous
- Autoriser le traffic à l'intérieur du projet msinfra
- Autoriser twitter microservice a accéder l'api twitter
- …

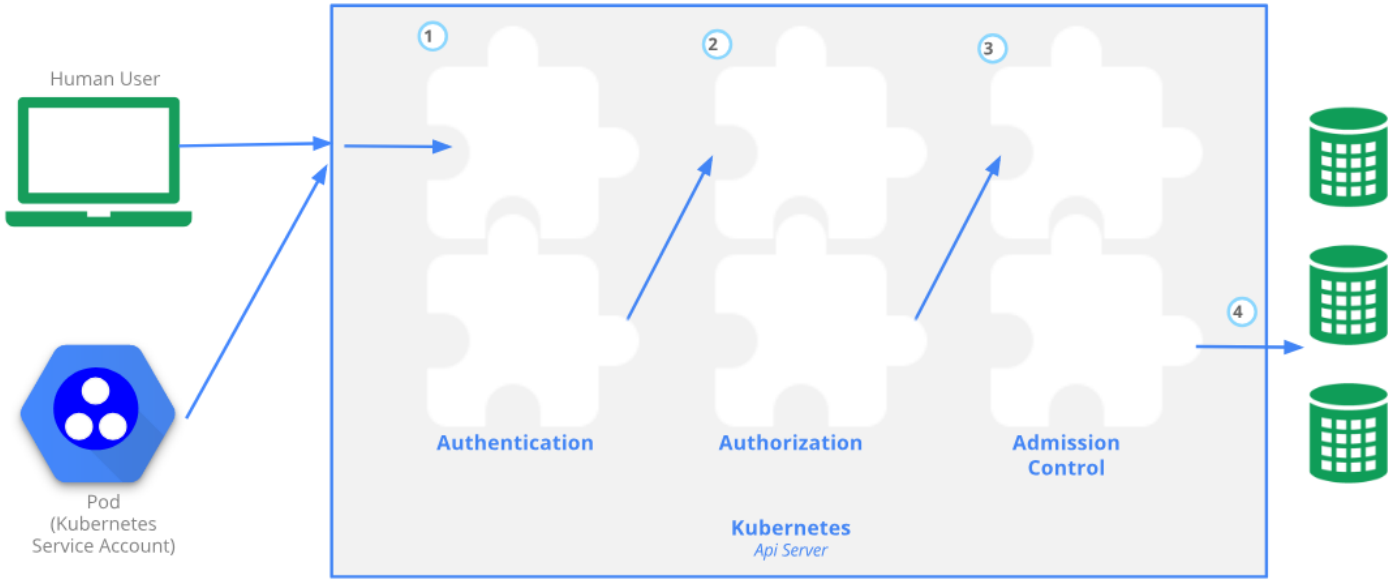
Sécuriser le Traffic: CNI

Recommandations:

- Choisir un CNIs qui permet de :
  - Chiffrer les communications (selon le besoin)
  - Implémenter des networkpolicies
- Choisir la bonne taille des packets (MTU size)
- Prendre en considération la consommation de ressources (CPU & RAM) lors du choix d'un CNI

Sécuriser le Control-Plane

Sécuriser le Control-Plane : Mécanismes



Sécuriser le control-plane : Authentification

Risques

- Récupérer des Credentials (Tokens) et les utiliser pour attaquer le serveur d'api
- Intercepter un certificat et l'utiliser pour s'authentifier sur d'autres services (Etcd)

Remédiation

- Activer au moins deux modes d'authentification (certificats x509 et Tokens openID)
- Désactiver l'accès anonyme au cluster
- Désactiver l'accès non authentifié à l'api :
  - Au niveau de la communication avec les kubelets
  - Au niveau de la communication avec le cluster Etcd
- Rotation des certificats + Strong Cryptographic Ciphers
- Utiliser un unique Certificate Authority pour l'Etcd (différente que celle de k8s)

Sécuriser le Control-Plane : Autorisation

Risques

- Accéder non autorisé sur des services/espace de nommage(namespaces)
- Accès privilégiés sur des ressources confidentiels
- Perte de données accidentelles

Remédiation

- **Mettre en place des politiques RBAC** :
  - Restreindre les kubelets a lire que les objets dont ils ont besoin
  - Création de teams, users et les affecter à des espace de nommage(namespaces) spécifique…
- Désactiver toutes les autorisations attribués aux requêtes (activer par défaut)
- Utiliser le role *cluster-admin* qu'en cas de besoin

Sécuriser le Control-Plane : Admission Control

- L'admission control permet d'avoir un filtrage plus granulaire sur les requêtes après l'authentification et l'autorisation

Risques:

- Attaques dos
- Accès non autorisé a des images privés

Remédiation:

- Exemple de plugins d'admission control:
  - *Eventratelimit* : limite le nombre d'évenements que peut accepter le serveur d'api en un lapse de temps
  - *Alwayspullimages: force les nouveaux pods a faire le pull de l'image a chaque fois*

Sécuriser le Control-Plane : Admission Control 2/2

- *Serviceaccount :* permet l'automatisation de la gestion des service accounts (voir section: sécuriser le control-plane : service account)
- *Namespacelifecycle :* garantit que les objets nouvellement créés ne peuvent pas être dans un/des espace de nommage(namespaces) inexistants ou en cours de terminaison .
- Podsecuritypolicy : permet de faire un contrôle granulaire sur les actions/les données qu'un pod peut exécuter/accéder(voir section:securiser podsecurity policies)
- *Noderestriction : restreint l'impact du kubelet sur les nodes et les pods (voir section:sécuriser le kubelet)*

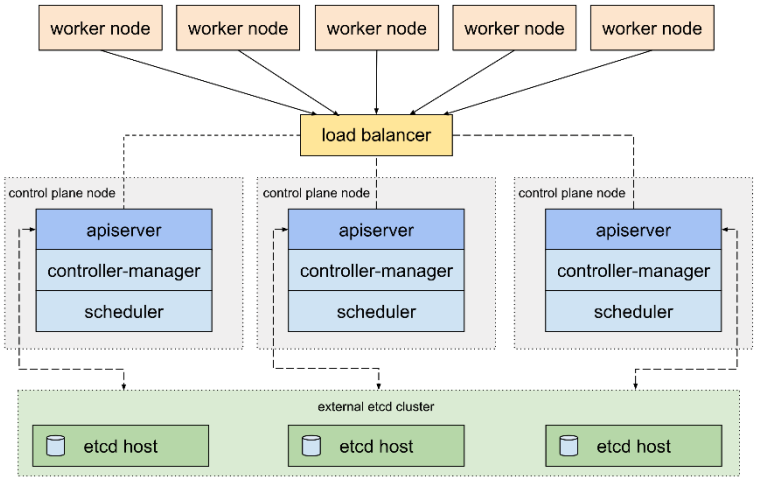
Sécuriser le Control-Plane : le cluster Etcd

Risques

- Un accès au cluster etcd c'est avoir un accès root sur tout le cluster
- Ajouter / supprimer des pods
- Modifier la configuration du cluster

Remédiation

- Restreindre les accès au cluster etcd
- Chiffrer les données sur l'etcd (encryption at REST)
- Rotation de la clé de déchiffrement



Sécuriser le control-plane : Service Account

- Les service account sont utilisés pour que les conteneurs d'un Pod puissent communiquer avec le serveur d'api directement

Recommandations:

- Utiliser un compte individuel pour chaque Controller (couplé avec RBAC)
- Ne pas utiliser le service account par défaut
- Au niveau du serveur API: vérifier le service account Token avant de valider l'authentification Token
- Automatisation de la gestion des service account

Sécuriser les composants internes

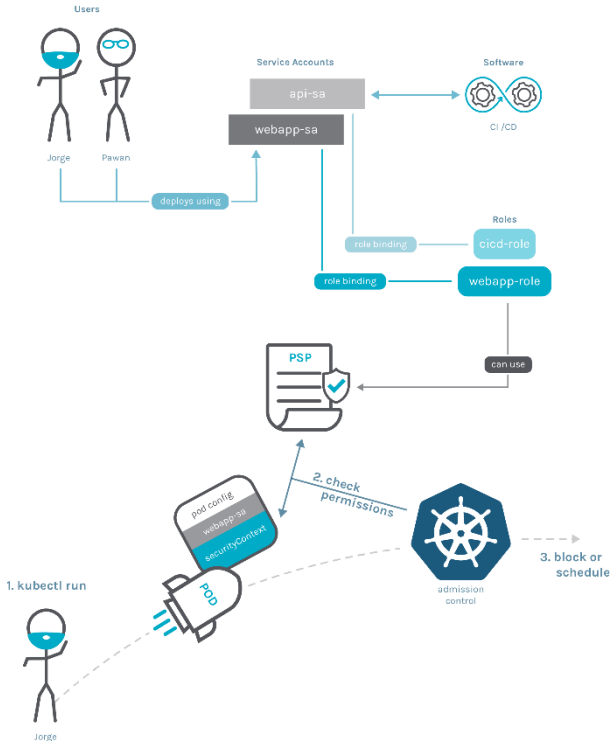
Sécurisation des composants internes

Composants à sécuriser :

- Images des conteneurs
- Pods
- Kubelets
- Secrets

> IT DOESN'T MATTER HOW MANY LOCKS ARE ON YOUR DOOR IF YOUR WINDOW IS OPEN

Sécuriser les Pods: PodSecurity Policies



Risques

- Elévation de privilèges(accés root sur le node)
- Ecoute de traffic circulant dans le node
- Accès à des données d'autres conteneurs

Remédiation

- Lancer les pods **en mode non-root** avec un système de fichiers root en Lecture-Seule
- Désactiver le permissions de modifier les security contexts .
- Restreindre le partage de espace de nommage (hostpid, hostipc, hostnetwork…)
- Restreindre les capabilities du pod

Sécuriser les kubelets

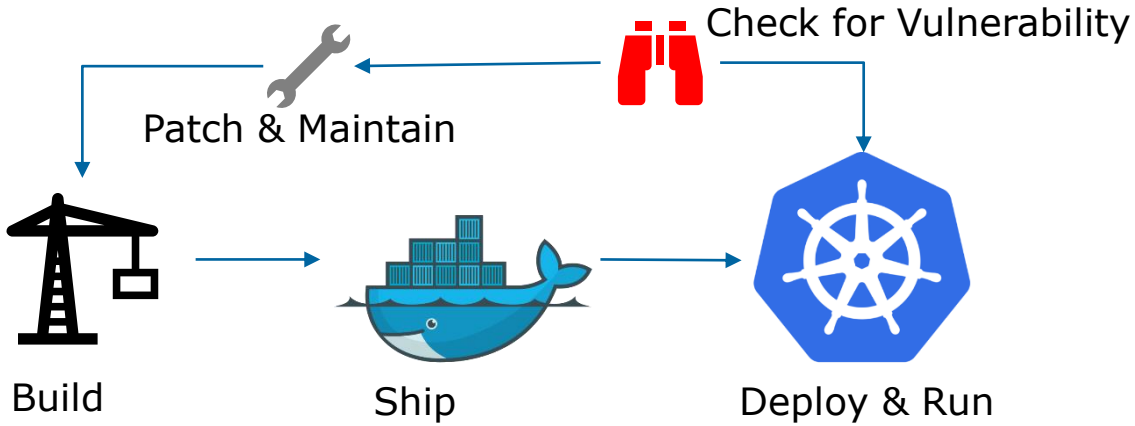
Risques

- Vol de données depuis n'importe quel pod du cluster
- Elévation de privilèges

Remédiation

- Désactiver les requêtes anonyme
- Activer l'autorisation explicite sur toutes les requêtes (RBAC, node)
- Désactiver le Lecture-Seule port
- Rotation des certificats

Sécuriser les images 1/2



Recommandations

- S'assurer que l'image source est certifié
- Scan de vulnérabilités à la fin du build
- Adopter les bonnes pratiques de développement d'une image de conteneur

Sécuriser les images 2/2

- Refuser les images de sources inconnues
- Vérifier l'intégrité de l'image avant le déploiement
- vérification réguliére de vulnérabilités sur les images utilisés
- Eviter la mise à jour directe dans l'image du conteneur
- Adopter une approche DevSecOps avec une automatisation des tâches via des pipelines

Sécuriser les secrets

Risques (Secret intégré à kubernetes)

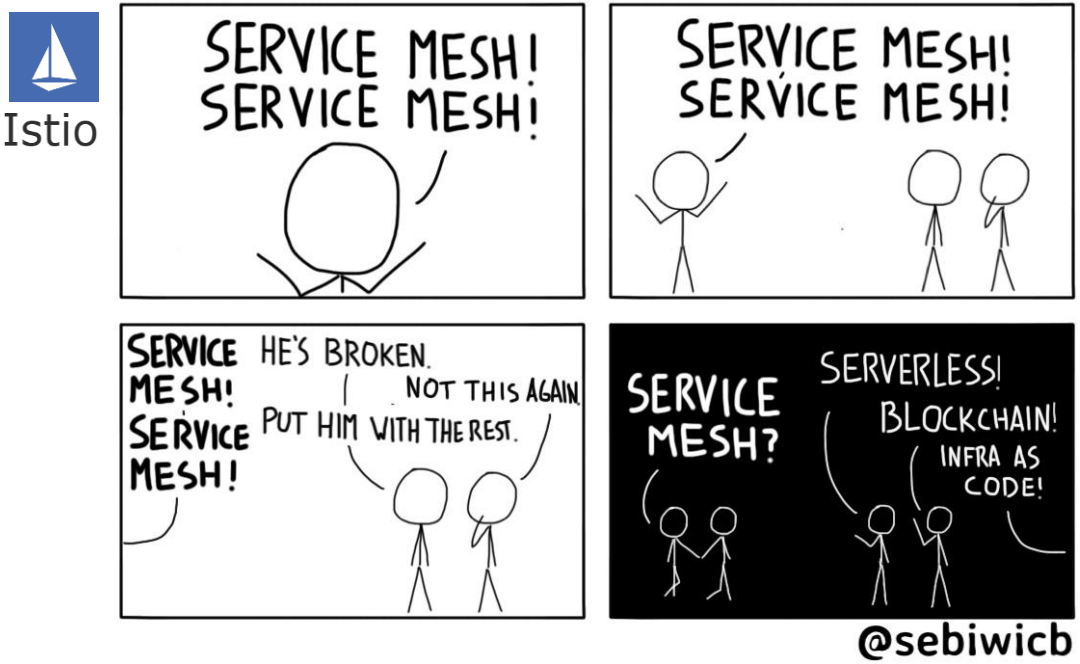
- Les secrets sont stockés en clair (pas chiffré)
- Un utilisateur qui consomme un secret peut le voir
- Dans le cas d'un root exploit, l'attaquant peut avoir tout les secrets
- Pas d'historisation des secrets
- Pas de contrôle granulaire sur les secrets

Recommandations

- L'utilisation de la ressource `Secret` intégré à kubernetes **n'est pas recommandée**
- L'utilisation d'une tierce partie est fortement recommandée (AWS secrets manager, google cloud platform KMS and azure key vault, hashicorp vault ..)

Perspectives

Perspectives

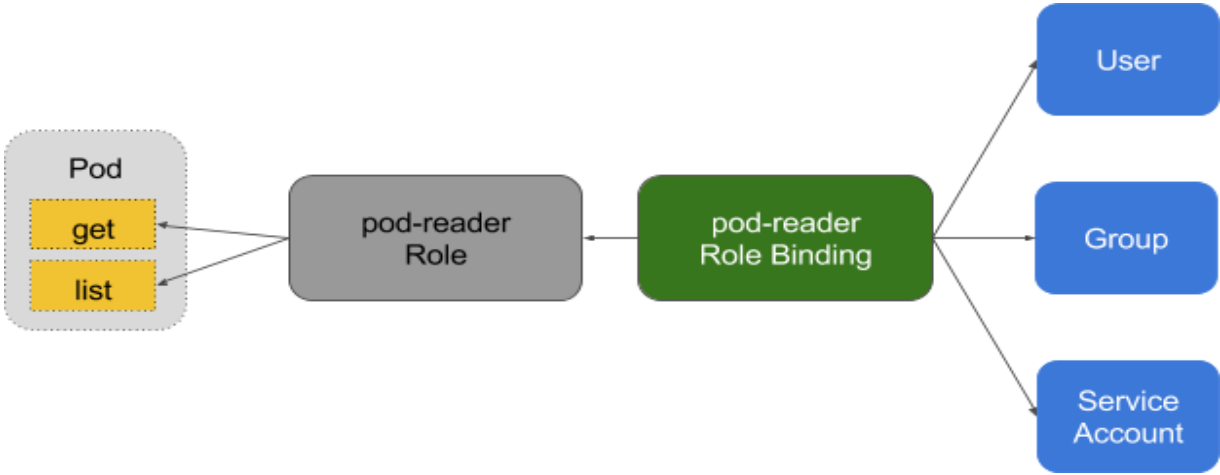


7

Appendix

RBAC

- Role-based access control provides fine-grained policy management for user access to resources, such as access to namespaces.



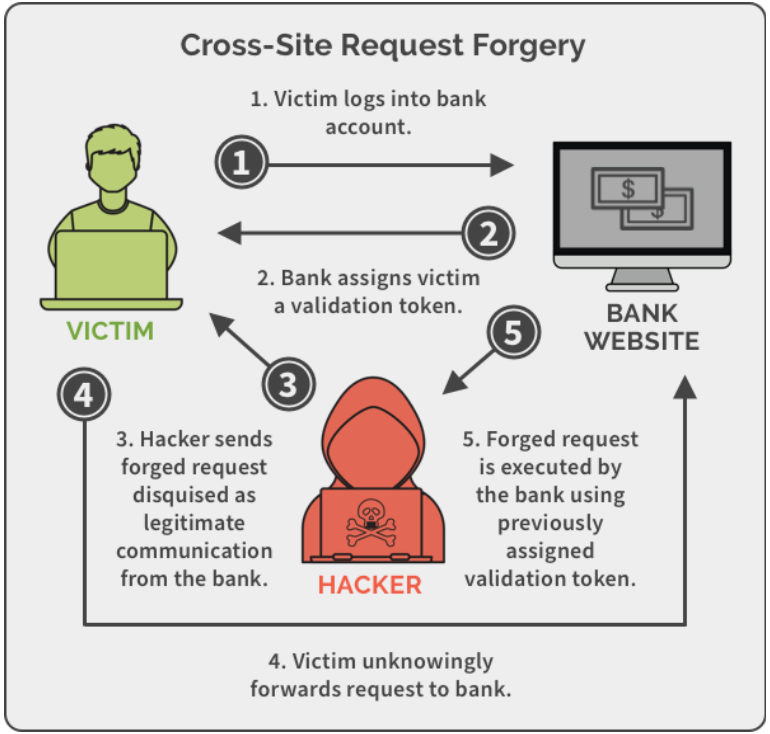
Strong Cryptographic Ciphers

The set of cryptographic ciphers currently considered secure is the following:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_AES_128_GCM_SHA256

Cross-site Request forgery

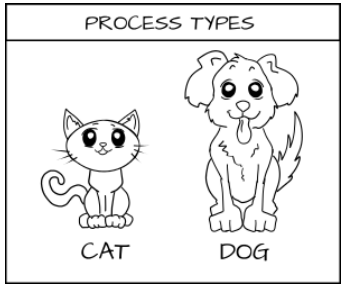
- Connu sous le nom de XSRF, sea surf et session riding
- Cross-site request forgery est une attaque qui force l'utilisateur final a exécuter des actions sur un site ou ils sont déjà authentifié sans leur consentement .
- Netflix, ING direct banking, youtube et mcafee secure avaient des vulnérabilités XSRF détecter sur leurs sites

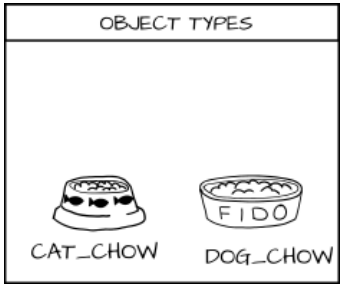


SELinux 1/2

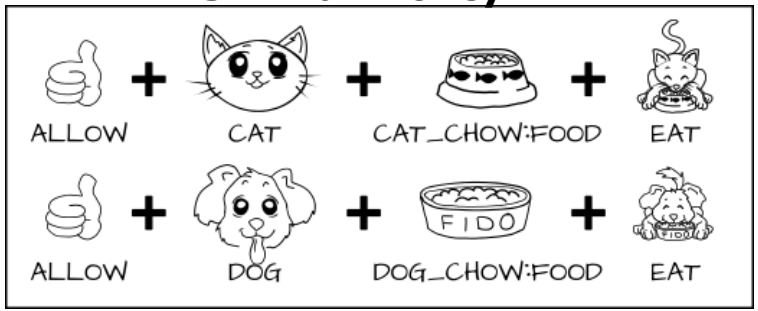
- Selinux (security-enhanced linux) est un module de sécurité développé initialement par la NSA et repris par redhat
- Intégrer de base sur le kernel de centos, RHEL, fedora
- Il permet d'implémenter des régles (security control policy) qui limite les accés sur les fichiers et surveiller l'execution des processus en cours d'execution
- Les policies sont sous forme de labels de type : user::role::type::level
- Selinux en mode strict refuse tout par défaut
- **Dans le contexte k8s :**
  - Selinux va assurer qu'un conteneur peut que lire et exécuter de /usr
  - Le(s) process du conteneur peut seulement ecrire dans le file system du conteneur

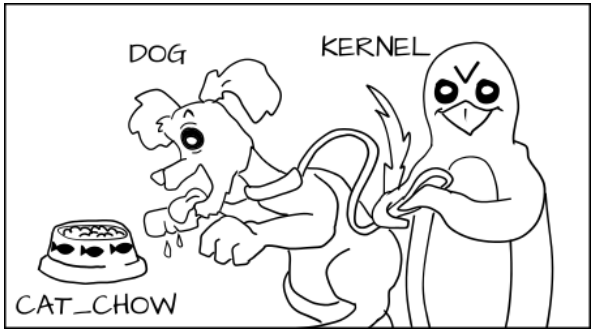
SELinux 2/2



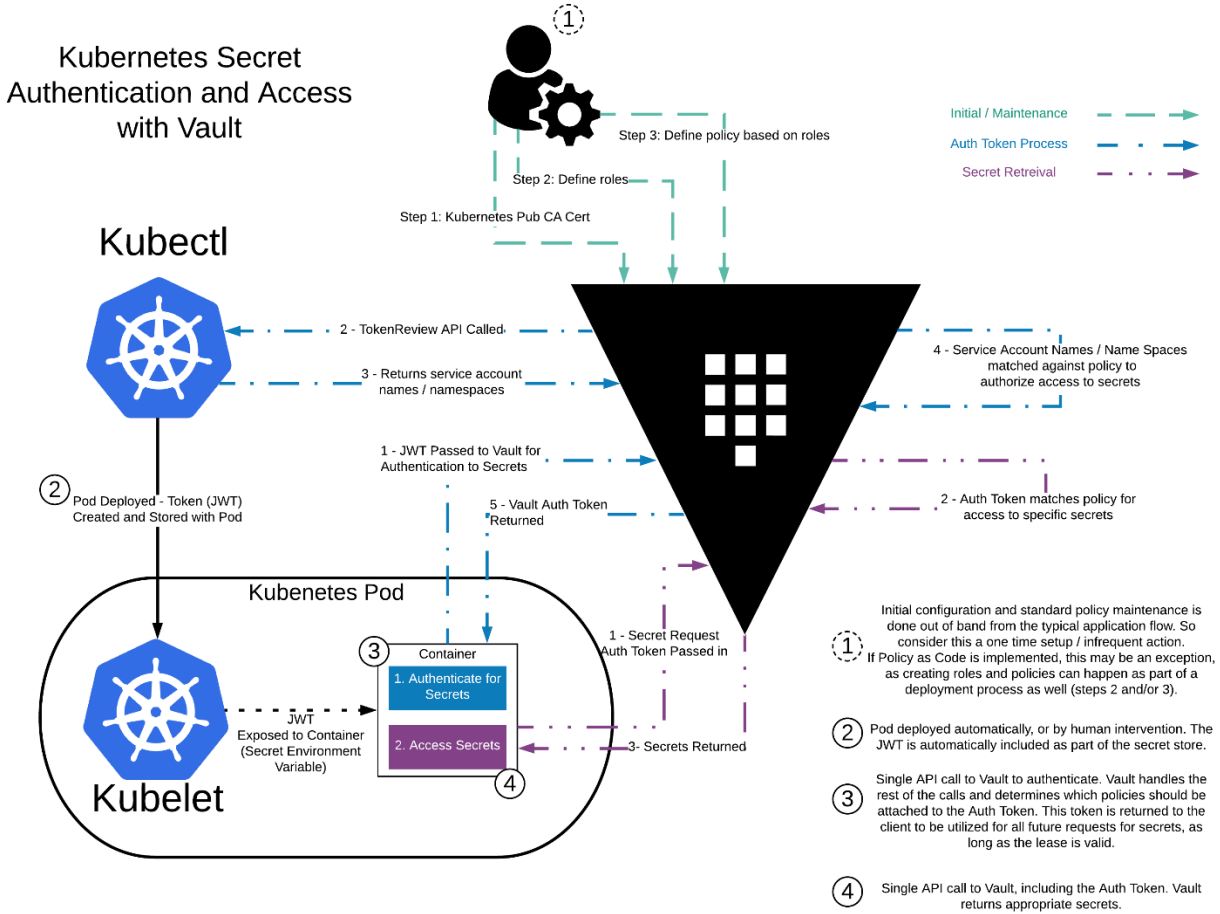


SELinux Policy:





SELinux assure que chaque process a le droit qu'a ce qui lui a été autorisé par les Security policies


Kubernetes Secret Authentication and Access with Vault
1
Step 3: Define policy based on roles
Initial / Maintenance
Auth Token Process
Secret Retreival
Step 2: Define roles
Step 1: Kubernetes Pub CA Cert
Kubectl
2 - TokenReview API Called
3 - Returns service account names / namespaces
4 - Service Account Names / Name Spaces matched against policy to authorize access to secrets
1 - JWT Passed to Vault for Authentication to Secrets
2 Pod Deployed - Token (JWT) Created and Stored with Pod
5 - Vault Auth Token Returned
2 - Auth Token matches policy for access to specific secrets
Kubenetes Pod
1 - Secret Request Auth Token Passed in
3
Container
1. Authenticate for Secrets
2. Access Secrets
4
JWT
Exposed to Container (Secret Environment Variable)
3- Secrets Returned
Kubelet
1 Initial configuration and standard policy maintenance is done out of band from the typical application flow. So consider this a one time setup / infrequent action. If Policy as Code is implemented, this may be an exception, as creating roles and policies can happen as part of a deployment process as well (steps 2 and/or 3).
2 Pod deployed automatically, or by human intervention. The JWT is automatically included as part of the secret store.
3 Single API call to Vault to authenticate. Vault handles the rest of the calls and determines which policies should be attached to the Auth Token. This token is returned to the client to be utilized for all future requests for secrets, as long as the lease is valid.
4 Single API call to Vault, including the Auth Token. Vault returns appropriate secrets.

Thank you

For more information please contact:
M+ 33 6 18 47 95 46
mohamed.elajroud@atos.net

Atos