La sécurité dans Kubernetes

Réflexions et meilleures pratiques en matière de protection des charges de travail sensibles

vmware

Table des matières

Au minimum, vous devez mettre en place un système de contrôle des accès basé sur les rôles (RBAC) dans votre cluster. Le RBAC, qui est activé par défaut dans la plupart des programmes d'installation récents, fournit un cadre pour la mise en œuvre du principe du moindre privilège pour les utilisateurs et les applications accédant à l'API Kubernetes.

Introduction

Les meilleures pratiques pour la sécurité Kubernetes évoluent rapidement. En effet avec les versions récentes, de nombreux problèmes de sécurité qui affectaient les premières versions de Kubernetes sont résolus par défaut. Cependant, comme pour tout système complexe, il existe encore des risques dont vous devez prendre conscience avant de lui confier vos données de production. Ce rapport résume les éléments les plus importants que vous devez avoir à l'esprit pour l'hébergement de charges de travail sensibles sur Kubernetes.

Les sujets abordés dans ce document visent à vous aider à comprendre les risques affectant potentiellement votre cluster. Le risque dans votre environnement dépend de votre modèle de menace et des types d'applications que vous exécutez dans votre cluster. Vous devrez réfléchir à la meilleure façon d'investir dans les contrôles et le renforcement de la sécurité en fonction de la sensibilité des données utilisées, du temps et des équipes dédiés à la sécurité, ainsi que des exigences aux normes de conformité dans votre secteur d'activité, telles que HIPAA, FISMA et PCI DSS.

Authentification du plan de contrôle et autorisations

Le plan de contrôle Kubernetes se compose du serveur d'API Kubernetes, du kubelet et d'autres composants de base qui planifient et exécutent ensemble les charges de travail dans votre cluster. L'accès au plan de contrôle, qui est similaire à l'accès SSH ou à l'accès physique à une machine, doit être rigoureusement contrôlé. Le système de contrôle d'accès de Kubernetes est assez souple pour personnaliser les contrôles d'accès et répondre ainsi aux besoins de votre environnement, mais cette flexibilité peut sembler complexe.

Au minimum, vous devez mettre en place un système de contrôle des accès basé sur les rôles (RBAC) dans votre cluster. Le RBAC, qui est activé par défaut dans la plupart des programmes d'installation récents, fournit un cadre pour la mise en œuvre du principe du moindre privilège pour les utilisateurs et les applications accédant à l'API Kubernetes.

Pour tirer le meilleur parti du contrôle RBAC, vous devez disposer d'une configuration appropriée :

- Utilisez chaque composant avec les autorisations les plus restrictives qui permettent cependant d'exécuter les fonctionnalités attendues. La plupart des applications de votre cluster auront besoin d'un accès limité, voire d'aucun accès, à l'API Kubernetes. Les composants système tels qu'un contrôleur d'ingress ou un système de surveillance peuvent avoir besoin d'un accès plus étendu, mais peuvent souvent être limités à un accès en lecture seule ou à un accès dans un espace de noms spécifique. Les journaux d'audit de l'API Kubernetes sont un outil utile pour découvrir les API utilisées par telle ou telle application.
- Vérifiez que les composants de référence ne permettent pas à des utilisateurs dont les autorisations sont limitées d'élever leurs droits. Les exemples du tableau de bord Kubernetes et du démon Helm tiller méritent une attention particulière. Isolez ces composants à l'aide de contrôles d'authentification et d'autorisations au niveau de l'application ou de contrôles d'accès réseau pour empêcher tout accès non autorisé.

Pour les utilisateurs, envisagez d'intégrer l'authentification Kubernetes à votre système de gestion des identités existant. Kubernetes est fourni avec la possibilité de s'authentifier auprès de n'importe quel fournisseur OpenID Connect qui respecte les exigences de conformité, tel que GitHub ou Google. Vous pouvez également étendre le système d'authentification et d'autorisation de Kubernetes avec des plugins basés sur webhook pour créer une intégration personnalisée des identités.

Il faut également sécuriser l'accès à l'infrastructure sous-jacente à Kubernetes, en incluant par exemple l'accès SSH ou le contrôle d'accès du fournisseur de Cloud, tel que AWS IAM. Ces systèmes contournent le système de contrôle d'accès de Kubernetes et constituent donc un point faible potentiel. Par exemple, un cybercriminel disposant d'un accès SSH à l'hôte Kubernetes etcd pourrait, avec une intention malveillante, insérer des définitions de pods pour exécuter arbitrairement du code ailleurs sur le cluster.

L'une des solutions au problème des contrôles d'accès réseau est l'authentification forte au niveau de la couche applicative, comme l'authentification TLS réciproque. L'identité d'une application de chiffrement est puissante car elle permet d'exprimer efficacement cette identité au-delà des limites du réseau.

Contrôle d'accès réseau

De nombreuses applications existantes supposent que l'accès au niveau du réseau implique un niveau d'autorisation. Même si vos applications s'appuient sur une authentification et une autorisation fortes au niveau de la couche applicative, le contrôle d'accès au niveau du réseau garantit une défense renforcée. Il assure une solide protection contre les vulnérabilités liées à l'autorisation préalable, telles que la vulnérabilité Heartbleed (CVE-2014-0160) dans OpenSSL.

Restriction du trafic interpod

Kubernetes fournit des types de données de base puissants pour spécifier les contrôles d'accès réseau entre les pods. Les règles réseau de Kubernetes peuvent limiter le trafic entrant vers un pod en fonction de l'espace de noms et des libellés du pod source, ainsi que de l'adresse IP pour le trafic issu de l'extérieur de votre cluster. Dans Kubernetes 1.8, la politique réseau peut également limiter le trafic sortant avec le même ensemble de sélecteurs. Un bon point de départ consiste à restreindre le trafic des ingress à l'espace de noms de votre application par défaut, comme décrit dans la documentation de la politique réseau de Kubernetes.

L'application de la politique réseau dépend de votre fournisseur d'interface réseau de conteneur (CNI). Certains fournisseurs de CNI mettent en œuvre ces contrôles. Sans eux, Kubernetes ne s'exécute pas : l'API accepte volontiers vos stratégies réseau, mais celles-ci ne sont pas appliquées. Si l'importance des contrôles d'accès réseau est cruciale pour vous, veillez à faire appel à un fournisseur, tel que Calico, qui implémente ces contrôles.

Contrôles du périmètre réseau

En fonction de votre environnement, vous souhaiterez peut-être appliquer certains contrôles des ingress et des egress à la frontière du réseau, en plus des contrôles au niveau des pods appliqués par Kubernetes. En particulier, dans le scénario où un assaillant a compromis l'une de vos applications et exploité le runtime ou le noyau de votre conteneur, vous ne pouvez pas vous fier au nœud pour appliquer les contrôles d'accès réseau.

Limites

Les contrôles d'accès réseau présentent certaines limites dans les environnements dynamiques tels que Kubernetes. Les difficultés sont les suivantes :

- Fédérer la politique réseau de Kubernetes sur plusieurs clusters
- Intégrer les contrôles de niveau réseau de Kubernetes et des contrôles réseau granulaires appliqués en dehors de la couche réseau du pod (par exemple, dans les groupes de sécurité AWS EC2)

Si vous êtes confronté à ces problèmes, demandez-vous si vous pouvez appliquer une politique de réseau moins précise et vous appuyer sur la couche applicative pour un contrôle d'accès granulaire.

Contrôle d'accès au niveau de la couche applicative

L'une des solutions au problème des contrôles d'accès réseau est l'authentification forte au niveau de la couche applicative, comme l'authentification TLS réciproque. L'identité d'une application de chiffrement est puissante car elle permet d'exprimer efficacement cette identité au-delà des limites du réseau.

Side-cars et l'approche du maillage des services

Même si les identités des applications sont provisionnées, le réaménagement des applications existantes pour mettre en œuvre l'authentification TLS réciproque peut être frustrant, chronophage et source d'erreurs. Une solution consiste à mettre en œuvre l'autorisation L7 via le modèle side-car. Dans ce modèle, chaque application possède un démon de proxy adjacent qui authentifie et met fin aux connexions entrantes et authentifie les connexions sortantes de manière transparente. Kubernetes facilite ce processus en permettant à plusieurs conteneurs de fonctionner ensemble avec un réseau local partagé pour le pod. Lorsque ce modèle est utilisé à l'échelle de votre cluster, on parle de maillage de services. Istio est une mise en œuvre précoce de cette approche.

Un modèle efficace consiste à faire en sorte que la plupart des utilisateurs interagissent avec le cluster de production uniquement via un pipeline de déploiement.

Renforcement de la sécurité des nœuds et du runtime de conteneur

Vous devez envisager la protection de la limite entre le conteneur et l'hôte. Ce concept est important, même dans les environnements à locataire unique, car une vulnérabilité d'exécution de code à distance, telle que Shellshock (CVE-2014-6271) ou la vulnérabilité d'analyse YAML de Ruby (CVE-2013-0156), peut transformer votre charge de travail, par ailleurs fiable, en un agent malveillant. Sans un renforcement approprié, il est possible que cette simple vulnérabilité d'exécution de code à distance se transforme en une prise de contrôle de tout un nœud ou de l'intégralité d'un cluster.

Les runtimes de conteneur actuels ne fournissent pas le sandbox le plus puissant possible, mais vous pouvez prendre certaines mesures pour atténuer le risque de vulnérabilités de fuite de conteneur :

- Segmentez vos clusters Kubernetes en mettant en place des niveaux d'intégrité, un moyen simple mais très efficace de limiter votre exposition à ce type de vulnérabilité. Par exemple, vos environnements de développement et de test peuvent être hébergés dans un cluster différent de celui de votre environnement de production.
- Investissez dans des correctifs faciles à mettre en œuvre pour les hôtes et le noyau. Assurez-vous de disposer d'un moyen de tester les nouvelles mises à jour du système (par exemple, un environnement de préproduction) et que vos applications peuvent tolérer une mise à niveau en continu du cluster sans impacter la disponibilité des applications.

Kubernetes excelle dans l'orchestration de ces mises à niveau. Une fois que vous vous êtes préparés suffisamment pour laisser Kubernetes rééquilibrer dynamiquement les pods d'application, la gestion des correctifs au niveau des nœuds devient relativement facile. Vous pouvez automatiser la mise à niveau en continu qui « nettoie » correctement chaque nœud et le met à niveau sur place ou, dans un environnement IaaS (infrastructure sous forme de service), le remplace par un nouveau nœud. Les investissements dans ce domaine améliorent également votre résilience globale aux pannes qui affectent les nœuds :

- Exécutez vos applications en tant qu'utilisateur non-root. Root (UID 0) dans un conteneur Linux reste le même utilisateur que root sur le nœud. Une combinaison de mécanismes de sandbox limite les actions du code s'exécutant dans le conteneur, mais les futures vulnérabilités du noyau Linux sont davantage susceptibles d'être exploitées par un utilisateur root que par un utilisateur sans droits.
- Activez et configurez des modules de sécurité Linux supplémentaires, tels que SELinux et AppArmor. Ces outils vous permettent de mettre en place un sandbox plus restrictif pour des conteneurs spécifiques. Dans de nombreux cas leur utilité est indéniable, mais la création et la maintenance de configurations appropriées nécessite d'y consacrer du temps. Ils ne sont pas forcément adaptés à toutes les applications ou tous les environnements.

Application de règles d'isolation

Conformément aux politiques de sécurité des pods, un mécanisme basé sur des règles contrôle que les applications de votre cluster utilisent en bonne et due forme le sandbox des conteneurs. Par exemple, vous pouvez exiger que tous les pods d'un espace de noms spécifique s'exécutent en tant que non-root, ne montent pas de systèmes de fichiers hôtes, et n'utilisent pas de réseau hôte.

Gestion des correctifs et pipelines de déploiement CI/CD

Un modèle efficace consiste à faire en sorte que la plupart des utilisateurs interagissent avec le cluster de production uniquement via un pipeline de déploiement. Ce pipeline comprend un ou plusieurs systèmes automatisés qui gèrent la création de code dans une image de conteneur, l'exécution de tests d'unité et d'intégration, ainsi que d'autres étapes de validation telles que la mise en pause pour une approbation manuelle. Selon vos besoins, les développeurs peuvent toujours avoir un accès direct en lecture seule à l'API Kubernetes ou disposer d'un moyen d'escalade au niveau des pods en cas d'incident.

La clé pour remédier aux vulnérabilités des images de conteneurs est également un puissant pipeline de déploiement d'applications. Des outils tels que Clair permettent d'identifier les vulnérabilités connues dans les bibliothèques et les packages que vous utilisez, mais pour publier les correctifs en temps voulu, vous avez besoin d'une méthode fiable et automatisée pour recréer et tester les versions corrigées du conteneur.

La journalisation constitue le socle de la surveillance de la sécurité. Généralement, vous devez capturer les journaux de l'application, les journaux au niveau de l'hôte, les journaux d'audit de l'API Kubernetes et les journaux du fournisseur de Cloud (le cas échéant). Des modèles bien établis permettent d'agréger les journaux sur les configurations de cluster courantes.

Limitation des fortes variations

Les clusters Kubernetes sains sont des environnements dynamiques. De nouvelles versions des applications sont déployées, des nœuds disparaissent au profit de mises à niveau du noyau, les déploiements sont modulés à la hausse ou à la baisse, et (espérons-le) les utilisateurs de votre application ne s'en aperçoivent jamais. En pratique, faire fonctionner tout cela exige une certaine diligence, ce qui est essentiel pour tirer parti de tous les avantages de Kubernetes.

Le budget d'interruption de pod est un outil susceptible de limiter le chaos introduit dans votre cluster. Il est utile lorsque vous disposez de plusieurs systèmes automatisés et que vous souhaitez garantir que leurs interactions ne sont pas dommageables. Par exemple, un bogue au niveau de l'application peut neutraliser temporairement certains pods de votre application. Un budget d'interruption de pod permettrait qu'une mise à niveau automatisée des nœuds en continu ne mette pas fin aux copies saines restantes de votre application.

Création de conteneurs avec des droits en nombre excessif

Le montage du socket de contrôle Docker */var/run/docker.sock* dans un conteneur pendant un processus de développement est un anti-modèle spécifique à Docker à éviter dans votre pipeline de création. L'accès à ce socket est équivalent à l'accès root sur l'hôte, ce qui signifie que n'importe quel build en cours d'exécution est susceptible de compromettre le nœud. Ce scénario se vérifie notamment si votre système de compilation exécute des builds avant la phase de révision manuelle du code (un modèle courant).

Gestion des secrets

Kubernetes dispose d'un socle pour la gestion des secrets d'application, appelé à bon escient « secret ». Généralement, les applications ont besoin de secrets pour deux raisons majeures :

1. Elles doivent avoir accès à des informations d'authentification qui prouvent leur identité auprès d'un autre système (par exemple, un mot de passe de base de données ou un jeton d'API tiers).
2. Elles ont besoin d'un secret cryptographique pour certaines opérations intrinsèques (par exemple, une clé de signature HMAC pour l'émission de cookies HTTP signés).

Secrets d'identité

Pour le premier cas d'usage, suivez les mesures prises dans le cadre du projet SPIFFE (Secure Production Identity Framework For Everyone) et du groupe de travail sur l'identité des conteneurs (Container Identity) visant à créer une solution à long terme pour le provisionnement dynamique d'identités d'applications uniques. À court terme, il n'existe pas de meilleures pratiques bien établies dans ce domaine, mais certains utilisateurs ont réussi l'intégration avec les workflows de provisionnement des certificats existants dans le cadre d'un pipeline d'intégration/déploiement continus (CI/CD). Des solutions Kubernetes simples telles que cert-manager sont également adaptées à votre cas d'usage.

Secrets non identitaires

Les meilleures pratiques évoluent encore pour le deuxième cas d'usage, mais de nombreux utilisateurs s'intègrent à des systèmes tels que Vault qui effectuent des opérations de chiffrement dans le cadre d'un service centralisé. Assurez-vous de comprendre toute la chaîne d'attestations impliquée dans l'authentification auprès d'un tel système, car elles dépendent souvent encore des ressources secrètes Kubernetes (une étape de la chaîne). Par exemple, le back-end d'autorisation Vault de Kubernetes authentifie les pods à l'aide d'un jeton de compte de service Kubernetes, mais ce jeton est stocké en tant qu'objet secret avant d'être injecté dans le pod. Ce modèle exige également que vous fassiez confiance à Vault pour ne pas reproduire votre jeton et prendre l'identité du pod auprès de l'API Kubernetes.

Risques liés aux secrets Kubernetes

Malheureusement, les secrets de Kubernetes s'accompagnent de certaines réserves. Le plus important est que de nombreux composants couramment utilisés, tels que les contrôleurs des ingress, requièrent actuellement une autorisation pour lire tous les secrets de votre cluster. Par défaut, les secrets ne sont pas chiffrés lorsqu'ils sont inactifs. Le support alpha du chiffrement est disponible dans Kubernetes 1.7, mais n'est pas encore recommandé pour une utilisation en production. Une solution consiste à utiliser le chiffrement au niveau du volume (par exemple, dm-crypt ou le chiffrement des volumes du fournisseur de Cloud) pour vos volumes de données etcd.

La méthode de sécurisation de votre cluster Kubernetes dépend en partie des ressources disponibles et des besoins des applications. Considérez chaque élément par rapport aux conditions globales de sécurité et prenez le temps d'évaluer son importance pour satisfaire l'ensemble de vos besoins.

EN SAVOIR PLUS SUR LA TECHNOLOGIE CLOUD NATIVE SELON VMWARE

Pour en savoir plus la manière dont les solutions VMware peuvent vous aider à créer, exécuter et gérer des applications Cloud, rendez-vous sur *https://cloud.vmware.com/*.

Surveillance et audit de la sécurité

Le niveau et le type de fonctions de surveillance de la sécurité adaptés à votre cluster dépendent largement du temps et des ressources que vous pouvez consacrer au traitement des alertes et au contrôle de l'environnement. En règle générale, il est déconseillé de consacrer du temps à la création de systèmes de surveillance de la sécurité que vous n'avez pas le temps d'entretenir et d'adapter. Commencez par les workflows en temps réel (basés sur les alertes) et périodiques (audit) de l'analyste ou de l'opérateur que vous voulez activer, et développez la plate-forme de surveillance dont vous avez besoin pour activer ces workflows.

Journalisation

La journalisation constitue le socle de la surveillance de la sécurité. Généralement, vous devez capturer les journaux de l'application, les journaux au niveau de l'hôte, les journaux d'audit de l'API Kubernetes et les journaux du fournisseur de Cloud (le cas échéant). Des modèles bien établis permettent d'agréger les journaux sur les configurations de cluster courantes.

À des fins d'audit de sécurité, envisagez de transférer vos journaux vers un emplacement externe avec un accès en mode « append-only » depuis votre cluster. Par exemple, sur AWS, vous pouvez créer un compartiment S3 dans un compte AWS isolé et octroyer un accès en mode append-only à l'agrégateur de journaux de votre cluster. Cela garantit que vos journaux ne peuvent pas être altérés, même dans le cas d'une compromission totale du cluster.

Surveillance du réseau

Les outils de surveillance de la sécurité basés sur le réseau, tels que les systèmes de détection d'intrusion (IDS) et les pare-feux d'application Web, sont opérationnels presque immédiatement, mais leur bon fonctionnement demande quelques efforts. Le principal obstacle est que de nombreux outils s'attendent à ce que les adresses IP fournissent un contexte utile pour les événements. Pour intégrer ces outils avec Kubernetes, pensez à compléter les événements collectés avec les métadonnées de l'espace de noms, du nom de pod et du libellé du pod de Kubernetes. Il s'agit d'un contexte de l'événement utile que vous pouvez utiliser à des fins d'alerte ou de révision manuelle, et qui peut rendre ces outils traditionnels encore plus puissants dans votre cluster que dans un environnement traditionnel. Certains outils de surveillance peuvent déjà collecter les métadonnées Kubernetes, mais vous pouvez également enrichir le code (événements personnalisés) pour ajouter ce type d'intégration de métadonnées à ceux qui ne le font pas.

Surveillance des événements de l'hôte

Il est également possible d'exécuter un IDS basé sur l'hôte, pour la surveillance de l'intégrité des fichiers et la journalisation des appels système Linux (par exemple, auditd), directement avec Kubernetes, mais les résultats sont difficiles à gérer car la charge de travail exécutée sur un nœud spécifique varie d'heure en heure à mesure que les applications se déploient et que Kubernetes orchestre les pods.

Pour donner un sens aux événements basés sur l'hôte, vous devrez à nouveau envisager d'étendre vos outils existants de manière à intégrer les métadonnées des pods ou des conteneurs Kubernetes dans le contexte des événements capturés. Les systèmes plus récents tels que Sysdig Falco incluent ce contexte prêt à l'emploi.

Quelles sont les étapes suivantes ?

La méthode de sécurisation de votre cluster Kubernetes dépend en partie des ressources disponibles et des besoins des applications. Considérez chaque élément par rapport aux conditions globales de sécurité et prenez le temps d'évaluer son importance pour satisfaire l'ensemble de vos besoins. De manière très générale, certaines de nos recommandations s'inscrivent parfaitement dans le cadre de meilleures pratiques plus étendues en matière de déploiement, par exemple :

- Pipeline de déploiement automatisé et planificateur. Ils vous permettent de simplifier la gestion des correctifs de l'hôte et des applications avec des mises à niveau en continu qui sont intégrées dans votre cycle de développement global.
- Contrôles d'accès intégrés à des niveaux appropriés.
- Journalisation et surveillance intégrées. Vous consignez et surveillez les performances et la fiabilité ; le support des événements de sécurité et des métadonnées de pods n'est pas chose aisée, mais il est essentiel. Vos besoins spécifiques déterminent ce qu'il faut surveiller exactement.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com
VMware Global Inc. Tour Franklin, 100-101 Quartier Boieldieu, 92042 Paris La Défense Cedex, France Tél. +33 1 47 62 79 00 www.vmware.com/fr