AYI NEDJIMI Consultants

Container Security : Docker et Runtime Protection Avancée

Catégorie : Cloud Security Lecture : 8 min Publié le : 12/03/2026 Auteur : Ayi NEDJIMI

Guide sécurité conteneurs Docker : durcissement images, scan vulnérabilités Trivy, protection runtime Falco Tetragon, registries Harbor et signature.

Les conteneurs Docker ont révolutionné le déploiement applicatif en apportant la portabilité, la reproductibilité et l'isolation des workloads. Cette technologie, devenue omniprésente dans les architectures modernes, introduit cependant des défis de sécurité spécifiques que les approches traditionnelles de protection des serveurs ne couvrent pas. La surface d'attaque des conteneurs s'étend des images et leurs dépendances au runtime Docker, en passant par l'orchestrateur, le registre d'images et le pipeline de build. En 2026, les attaques ciblant les conteneurs ont considérablement augmenté en sophistication, exploitant des vulnérabilités dans les images de base, les configurations Docker permissives et les failles dans la chaîne d'approvisionnement logicielle. Ce guide couvre l'ensemble de la chaîne de sécurité des conteneurs, depuis la création d'images sécurisées jusqu'à la protection runtime en production, en s'appuyant sur des outils open-source et commerciaux éprouvés dans des environnements réglementés et hautement sensibles.

Résumé exécutif

Guide complet de sécurité des conteneurs : durcissement des images Docker, scan de vulnérabilités CI/CD, protection runtime, gestion des registries et bonnes pratiques de configuration pour les environnements de production.

Retour d'expérience : lors d'un audit de la chaîne CI/CD d'un opérateur télécom, nous avons découvert que 78 % des images Docker en production contenaient des vulnérabilités critiques dont certaines dataient de plus de deux ans. L'image de base officielle Node.js utilisée incluait 134 CVE dont 12 critiques. La mise en place d'un pipeline de scan avec Trivy, combinée à la migration vers des images distroless, a réduit le nombre de vulnérabilités de 89 % en quatre semaines, sans impact sur le fonctionnement des applications. Face à la complexité croissante des environnements cloud hybrides et multi-cloud, les organisations doivent adopter des stratégies de sécurité adaptées aux spécificités de chaque fournisseur tout en maintenant une cohérence globale. Les équipes sécurité sont confrontées à des défis inédits : surfaces d'attaque dynamiques, configurations éphémères, gestion des identités à grande échelle et conformité réglementaire multi-juridictionnelle. Ce guide technique présente les approches éprouvées en environnement de production, les erreurs fréquentes à éviter et les stratégies de durcissement prioritaires. Chaque recommandation est issue de retours d'expérience concrets en entreprise et a été validée sur des architectures cloud de production à grande échelle.

Création d'images Docker sécurisées

La sécurité des conteneurs commence dès la rédaction du *Dockerfile*. Le choix de l'**image de base** détermine la surface d'attaque initiale : les images Alpine Linux, les images distroless de Google et les images scratch offrent une empreinte minimale comparée aux images Ubuntu ou Debian complètes. Les **builds multi-stage** permettent de séparer l'environnement de compilation de l'image finale, éliminant les outils de build, les fichiers temporaires et les dépendances de développement qui augmentent inutilement la surface d'attaque. Chaque instruction `RUN` doit être optimisée pour minimiser les couches et supprimer les caches de packages dans la même instruction.

Le principe du **moindre privilège** s'applique à l'utilisateur d'exécution du conteneur. L'instruction `USER` doit spécifier un utilisateur non-root dédié, créé avec un UID spécifique pour éviter les conflits. Les **capabilities Linux** doivent être restreintes au minimum via `--cap-drop=ALL` avec ajout sélectif des capabilities nécessaires. L'instruction `HEALTHCHECK` permet de détecter les conteneurs en état dégradé et de faciliter leur redémarrage automatique. Les *labels de sécurité* documentent l'image avec le mainteneur, la version, la date de build et le hash du commit source, facilitant la traçabilité. Les secrets ne doivent jamais être inclus dans les images via `COPY` ou `ENV`, mais injectés au runtime via des montages de volumes ou des variables d'environnement provenant d'un coffre-fort. Consultez les recommandations de AWS Security pour la sécurisation des conteneurs sur AWS ECS et EKS. Notre article sur Securiser Acces Microsoft 365 Mfa détaille les aspects spécifiques de la sécurité Kubernetes.

Scan de vulnérabilités dans le pipeline CI/CD

Le scan de vulnérabilités des images conteneurs doit être intégré à chaque étape du cycle de vie. Dans le **pipeline CI/CD**, le scan s'exécute après le build et avant le push vers le registre, bloquant les images qui ne respectent pas les critères de sécurité définis. **Trivy**, développé par Aqua Security, est le scanner open-source de référence avec sa couverture étendue des bases de vulnérabilités (NVD, Alpine SecDB, Red Hat OVAL, Debian Security Tracker) et sa capacité à scanner les images, les fichiers système, les dépendances applicatives et les misconfigurations. **Grype** d'Anchore offre une alternative performante avec une excellente intégration dans les workflows CI/CD.

La politique de scan doit définir des **seuils de blocage** progressifs : tolérance zéro pour les CVE critiques avec exploit connu, délai de remédiation de sept jours pour les critiques sans exploit, et trente jours pour les vulnérabilités hautes. Les *exceptions documentées* gèrent les cas où une vulnérabilité ne peut pas être corrigée immédiatement (dépendance indirecte sans correctif disponible) avec une justification technique et une date d'expiration. Le scan ne doit pas se limiter aux vulnérabilités OS mais couvrir les **dépendances applicatives** (npm, pip, Maven, Go modules) qui représentent une proportion croissante des vulnérabilités exploitées. L'intégration avec les plateformes d'intégration continue comme GitHub Actions, GitLab CI et Jenkins permet l'automatisation complète du workflow scan-décision-notification. Notre guide sur Secrets Sprawl Collecte Guide approfondit les stratégies de pipeline CI/CD sécurisé. Consultez CIS Benchmarks pour les standards de sécurité applicables aux conteneurs.

Protection runtime des conteneurs

Le scan d'images détecte les vulnérabilités connues au moment du build, mais ne protège pas contre les attaques zero-day, les comportements malveillants et les compromissions en cours d'exécution. La **protection runtime** surveille le comportement des conteneurs en temps réel pour détecter les anomalies. **Falco** intercepte les appels système via eBPF et les compare à des règles de détection pour identifier les comportements suspects : ouverture d'un shell dans un conteneur, lecture de fichiers sensibles (`/etc/shadow`, `/proc/`), communication réseau vers des destinations inattendues, modification de binaires système, montage de volumes non autorisés.

Sysdig Secure combine le scan d'images avec la protection runtime dans une plateforme commerciale unifiée. **Aqua Security** offre des capacités de *runtime enforcement* qui bloquent les comportements non autorisés plutôt que de simplement alerter. **NeuVector**, désormais open-source sous la fondation SUSE, propose une protection Layer 7 avec inspection du trafic réseau entre conteneurs. L'approche la plus avancée utilise le **profiling automatique** : l'outil apprend le comportement normal du conteneur pendant une période d'observation puis alerte ou bloque tout écart significatif. Cette approche réduit considérablement les faux positifs mais nécessite une période d'apprentissage stable. Consultez ANSSI pour les recommandations de sécurité runtime sur GCP. Notre article sur Migration Vmware Proxmox apporte des perspectives complémentaires sur la sécurité des workloads cloud.

Outil	Type	Forces	Licence
Trivy	Scanner images	Couverture CVE, vitesse, multi-format	Open source
Grype	Scanner images	Intégration CI/CD, SBOM natif	Open source
Falco	Runtime detection	eBPF, règles CNCF, communauté	Open source
Tetragon	Runtime enforcement	eBPF, blocage inline, performance	Open source
Cosign	Image signing	Sigstore, keyless signing	Open source
Harbor	Registry sécurisé	Scan intégré, RBAC, réplication	Open source

Registries sécurisés et signature d'images

Le **registre d'images** est un composant critique de la chaîne de sécurité des conteneurs. **Harbor**, projet CNCF, offre un registre privé avec scan de vulnérabilités intégré (Trivy), contrôle d'accès RBAC, réplication géographique et politiques d'immutabilité des tags. Les registres managés des cloud providers (ECR, ACR, Artifact Registry) s'intègrent nativement avec les services d'orchestration et proposent le scan automatique des images au push. La politique de rétention doit limiter le nombre de versions conservées pour réduire la surface d'attaque des images obsolètes.

La **signature d'images** avec *Cosign* (projet Sigstore) garantit l'authenticité et l'intégrité des images déployées. Le mode keyless de Cosign utilise des certificats éphémères liés à l'identité OIDC du signataire, éliminant la complexité de gestion des clés. L'**admission controller** Kubernetes vérifie la signature avant d'autoriser le déploiement d'une image, créant une chaîne

de confiance complète du build au runtime. La génération automatique de *SBOM* (Software Bill of Materials) avec **Syft** documente toutes les dépendances incluses dans l'image, facilitant la réponse aux nouvelles vulnérabilités par l'identification rapide des images affectées. Notre article sur Supply Chain Applicative détaille les aspects supply chain de la sécurité logicielle. Consultez AWS Security pour les pratiques de sécurité des registres ECR sur AWS.

Mon avis : la sécurité des conteneurs souffre d'un décalage entre la maturité des outils de scan (excellents) et l'adoption des protections runtime (insuffisante). Trop d'organisations se contentent de scanner les images sans surveiller le comportement en production, laissant un angle mort considérable. L'investissement dans une solution runtime comme Falco ou Tetragon devrait être considéré comme aussi fondamental que le scan de vulnérabilités, surtout dans les environnements exposés à internet.

Comment sécuriser les images Docker en production ?

La sécurisation des images Docker en production repose sur une chaîne de contrôles couvrant le build, le stockage et le déploiement. Au **build**, utilisez des images de base minimales (Alpine, distroless), appliquez les builds multi-stage pour éliminer les artefacts de compilation, exécutez le conteneur sous un utilisateur non-root avec des capabilities minimales, et ne stockez jamais de secrets dans l'image. Au **stockage**, utilisez un registre privé avec scan automatique au push, signez les images avec Cosign et appliquez des politiques d'immutabilité des tags pour empêcher la substitution d'images. Au **déploiement**, configurez un admission controller qui vérifie la signature et l'absence de vulnérabilités critiques avant d'autoriser l'exécution, utilisez des références par digest (`@sha256:...`) plutôt que par tag pour garantir l'immutabilité, et appliquez un profil seccomp restrictif. La maintenance continue implique la reconstruction régulière des images pour intégrer les correctifs de sécurité des images de base, avec un pipeline de rebuild automatique déclenché par les notifications de nouvelles CVE. Notre article sur Cloud Network Security Vpc Waf Ddos offre un guide complémentaire sur la sécurité des conteneurs dans le contexte cloud.

Pourquoi la sécurité runtime des conteneurs est-elle indispensable ?

La sécurité runtime comble un angle mort fondamental du scan statique des images. Le scan détecte les vulnérabilités connues et référencées dans les bases CVE au moment du build, mais plusieurs scénarios échappent à cette approche. Les **vulnérabilités zero-day** ne sont pas encore référencées et passent donc inaperçues au scan. Les **attaques comportementales** exploitent des fonctionnalités légitimes de manière malveillante (reverse shell via netcat installé, exfiltration DNS, cryptominage). Les **dérives de configuration** en runtime modifient le comportement du conteneur après son déploiement initial. Les **compromissions de la supply chain** injectent du code malveillant qui n'est pas détectable par un scan de vulnérabilités standard. La protection runtime fournit une couche de défense essentielle qui détecte ces scénarios en temps réel, réduisant considérablement le délai entre la compromission et la détection, qui peut être de plusieurs semaines voire plusieurs mois sans monitoring adéquat.

Quelles sont les vulnérabilités Docker les plus critiques ?

Les vulnérabilités Docker les plus dangereuses exploitent les mécanismes d'isolation des conteneurs pour obtenir un accès au système hôte. Les **conteneurs en mode privileged** désactivent toutes les protections d'isolation, permettant l'accès direct aux devices du noeud, la modification des règles iptables et le montage du système de fichiers hôte. Les **volumes hostPath** exposent des portions arbitraires du système de fichiers hôte dans le conteneur, incluant potentiellement le socket Docker (`/var/run/docker.sock`) qui permet le contrôle total du daemon Docker. Les **capabilities Linux excessives** comme `CAP_SYS_ADMIN` ou `CAP_NET_ADMIN` accordent des privilèges quasi équivalents à root. Les **images avec des packages vulnérables** non mis à jour exposent des CVE exploitables depuis le réseau ou le système de fichiers. Les **secrets codés en dur** dans les images (clés API, mots de passe, certificats) sont extractibles par simple inspection des couches Docker. Enfin, les *escape de conteneur* via des vulnérabilités dans runc ou containerd permettent l'exécution de code sur l'hôte depuis un conteneur non privilégié. Le CIS Benchmarks publie des benchmarks détaillés pour évaluer et corriger ces configurations Docker.

À retenir : la sécurité des conteneurs Docker couvre trois phases : build (images minimales, multi-stage, scan de vulnérabilités), ship (registres privés, signature avec Cosign, SBOM) et run (protection runtime Falco/Tetragon, Network Policies, profils seccomp). Une stratégie complète combine le shift-left dans le pipeline CI/CD avec la protection runtime en production pour une défense en profondeur.

Vos images Docker de production sont-elles scannées automatiquement à chaque build, ou déployez-vous encore des conteneurs sans connaître leur profil de vulnérabilités ?

Sources et références : CISA · Cloud Security Alliance

Perspectives et prochaines étapes

L'avenir de la sécurité des conteneurs est marqué par trois tendances majeures. L'adoption de WebAssembly (Wasm) comme alternative aux conteneurs pour certains workloads apporte une isolation plus forte par conception, avec une surface d'attaque réduite. La standardisation des SBOM via les formats SPDX et CycloneDX facilite la transparence de la supply chain logicielle et la réponse rapide aux nouvelles vulnérabilités. L'évolution des technologies eBPF rend la protection runtime plus performante et plus granulaire, permettant des politiques de sécurité au niveau des appels système individuels sans impact mesurable sur les performances. Les organisations qui investissent dès maintenant dans une chaîne de sécurité conteneur complète seront les mieux positionnées pour adopter ces évolutions technologiques.

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

ayinedjimi-consultants.fr · ayi@ayinedjimi-consultants.fr