Attestkeep

Ürün el kitabı

Kubernetes için container image güvenliği. Attestkeep image'ları tarar, imzalarını doğrular, bir pod'un başlayıp başlamayacağına admission anında karar verir, ekibinizin deny list'e koyduğunu reddeder ve her kararın imzalı kaydını tutar. Bu kitap ürünün cluster'a ne kurduğunu, nasıl işletildiğini ve ne kadara mal olduğunu anlatır.

Attestkeep 1.4.0 · Helm chart `oci://ghcr.io/attestkeep/charts/attestkeep` sürüm 1.4.0 · 29 Ekim 2026.

Kitaptaki ekranlar 29 Ekim 2026'da, konsolu 1.4.0 sürümünü gösteren, Kubernetes v1.35.1 üzerinde Enterprise Edition olarak çalışan ve Falco'nun bağlı olduğu bir kurulumdan, konsol dili Türkçe seçilerek alındı. Ek bu kurulumu ayrıntılı anlatıyor. Komutlar, ayarlar ve limitler docs.attestkeep.com'daki dokümantasyondan, chart'tan ve konsoldan alındı. Kitapta temsili hiçbir şey yok.

attestkeep.com · docs.attestkeep.com · support@attestkeep.com

İçindekiler

1. Attestkeep nedir

2. Cluster'da ne çalışır

3. Admission kararı nasıl verilir

4. Kurulum, yükseltme ve ilk açılış

5. Konsol

6. Admission policy'leri

7. Deny list

8. Falco'dan gelen runtime event'leri

9. Enforcement'ın çalıştığını görmek

10. Kanıt paketleri ve güven modeli

11. İşletme, SIEM aktarımı ve yedekleme

12. Lisans

13. Planlar ve fiyatlar

14. Güvenlik, destek ve muhatabınız

Ek. Bu kitabın dayandığı kaynaklar

1. Attestkeep nedir

Attestkeep bir Kubernetes operator'üdür. Workload'larınızın kullandığı image'ları tarar, imzalarını ve attestation'larını doğrular. API server bir pod için sorduğunda, yazdığınız policy'ye göre izin verir ya da reddeder. Her karar kaydedilir. Bu kayıtlardan, denetçinin kendi araçlarıyla doğrulayabileceği dönem raporu üretilir.

Her şey sizin cluster'ınızda çalışır. Operator, API, konsol ve veritabanı tek bir Helm release ile sizin seçtiğiniz namespace'e kurulur. Image adları, bulgular ve cluster topolojisi cluster'dan çıkmaz. Lisans kontrolü günde bir kez üç değer gönderir.

Neler yapar

- **Zafiyet taraması.** Trivy image'ın içinde gelir. Bulgular severity'ye göre, varsa düzeltilmiş sürümle listelenir.
- **İmza doğrulama.** Anahtarlı ya da keyless cosign doğrulaması. in-toto attestation'larında azami yaş kontrolü. 1.4.0'dan itibaren policy kabul ettiği public key'leri listeleyebilir, o zaman yalnız bu anahtarlardan birinin doğruladığı imza geçer.
- **Admission kontrolü.** Severity eşikleri, tag ve registry kuralları, digest zorunluluğu, imza ve attestation şartları, on beş workload hardening kontrolü, CISA'nın Known Exploited Vulnerabilities kataloğuna bakan kural, desteği bitmiş base OS kuralı, imzalayan kimliği de kontrol eden SLSA build provenance kuralı, üreticinin VEX beyanları ve privileged ya da dışarıya açık pod'lar için daha sıkı değerlendirme.
- **Deny list.** Bir image'ın, bir CVE'nin, bir paket sürümünün ya da bir dosyanın çalışmaması gerektiğine karar verildiğinde bunu tek bir kayıt söyler. Kayda break-glass'tan ve policy'nin bütün kurallarından önce bakılır, policy'si listeyi zorlayan her namespace'te. İkinci bir araç image'ların içindeki çalıştırılabilir dosyaları okur, böylece kaynaktan derlenmiş, paket kaydı olmayan bir runtime da bulunur. Çalışan bir workload kendiliğinden durdurulmaz. Bir yönetici onu elle, bir kez take down edebilir ve geri açabilir. Ayrıntısı 7. bölümde.
- **Süreli triage.** Bir riski gerekçe ve bitiş tarihiyle kabul edebilirsiniz. Olay anında break-glass, adı verilen image'ı içeri alır. İkisi de kayda geçer.
- **Kanıt paketi.** Her denetim dönemi için seçtiğiniz kontrol setine eşlenmiş tek, imzalı bir belge. Neyi kanıtlayamadığını da yazar.
- **Audit log.** Policy değişiklikleri, admission istisnaları, triage kararları, kanıt dışa aktarımları ve oturum açma olayları tek kayıtta.
- **Bildirimler.** Slack, Teams, e-posta ve genel webhook. Bildirim cluster'ın adını söyler, topolojisini söylemez.
- **Issue tracker'lar.** Yeni bir kritik bulgu, GitHub, GitLab ya da Jira'nızda image digest ve CVE başına bir issue açar. Issue kapanırken sebebi yorum olarak yazılır: düzeltildi, triage edildi ya da image kullanımdan kalktı. Remediation olarak yalnız "düzeltildi" sayılır.
- **Falco'dan runtime event'leri.** Kendi Falco'nuz event'lerini mutual TLS ile operator'e gönderir, doğrudan ya da Falcosidekick üzerinden. Her event geldiği image ve pod'la birlikte saklanır. Konsolda yazdığınız runtime policy'leri sonrasını belirler: yalnız kayıt, bildirim, issue ya da pod'u durdurmak. Runtime event'i admission kararını değiştirmez. Ayrıntısı 8. bölümde.
- **SIEM aktarımı.** Reddedilen admission'lar, deny list eşleşmeleri ve take down'lar, runtime event'leri, policy ve triage değişiklikleri, kanıt indirmeleri, başarısız girişler ve yeni kritik bulgular olduğu anda SIEM'inize gider: Splunk, Microsoft Sentinel, Elasticsearch, syslog ya da genel bir HTTP collector. Ayrıntısı 11. bölümde.
- **Single sign-on ve roller.** Google, GitHub, GitLab, Okta, Microsoft Entra ID, Keycloak ya da herhangi bir OIDC sağlayıcı, ücretli planlarda. Policy'yi kimin değiştirebileceğini roller belirler.

Neler yapmaz

Attestkeep bir runtime sensörü değildir. Çalışan container'daki syscall'ları, ağ trafiğini ya da process'leri izlemez. Bu tespiti Falco yapar, Attestkeep yalnız Falco'nun bildirdiğini alır. Attestkeep Falco'yu paketlemez, kurmaz, yapılandırmaz ve Falco'ya yazmaz.

Çalışan workload'ları kendi kendine durdurmaz. Deny list kaydı da enforce modundaki policy de yeni pod'ları reddeder. Zaten çalışanı durdurmak her zaman bir kişinin kararıdır, ya da sizin yazdığınız ve chart'ın izin verdiği bir runtime

policy'sinin.

Cloud posture aracı da değildir, cloud hesabınızı ve IAM policy'lerinizi okumaz. Terraform'u, manifest'leri ya da cluster yapılandırmasını bir benchmark'a göre taramaz. Zafiyetli bir fonksiyonun gerçekten çağrılıp çağrılmadığına bakmaz. Repository'lerinizi okumaz, dependency pull request'i açmaz. Bunların her biri için yanına ne konacağı docs.attestkeep.com'da yazılı.

Aynı kapıyı ücretsiz araçlarla kurmak

Kurulabilir. İyi açık kaynak admission policy motorları ve cluster içi tarayıcılar var. Tek ihtiyacınız `:latest`'i engellemek ve kritik CVE'de reddetmekse Attestkeep'e gerek yok. Böyle bir kurulumda genellikle eksik kalan, denetimde ve olay incelemesinde kullanılan kayıttır:

- **Bütünlüğü korunan karar kaydı.** Her admission kararı yazıldığı anda hash'lenir ve imzalı, zincirli checkpoint'lerle mühürlenir. Veritabanında tek bir karar değiştirilirse, sonraki kanıt belgesi o kaydı gösterir.
- **Kararla policy'nin bağı.** Her kayıt, o anki policy metninin hash'ini taşır.
- **Raporlanan boşluklar.** Webhook'un çalışmadığı ya da doğrulanamadığı süreler kaydedilir ve kanıta basılır. Reconciliation taraması bu sürelerde içeri ne girdiğini listeler.
- **Tek ürün.** Tarama, imza kontrolü, admission, deny list, istisnalar ve kanıt paketi birlikte bakımı yapılan tek bir üründe durur. Aradaki bağlantı kodunu ekibinizin taşıması gerekmez.

2. Cluster'da ne çalışır

Admission webhook her pod hakkında söz sahibi olur. Bunun karşılığında cluster'a kurulanlar şunlar.

Nesne	Varsayılan	Görevi
Deployment `attestkeep`	2 replica, node'lar arasında tercihli anti-affinity, PodDisruptionBudget minAvailable 1	Operator: admission webhook (8443), konsol ve API (8080), metrikler (9090). `integrations.falco.enabled` açıksa Falco için integrations portu (8444). Controller işleri PostgreSQL advisory lock ile sıraya girer, replica sayısı leader election olmadan artırılabilir. 1.4.0'dan itibaren kapanan replica cevap vermeyi bırakmadan önce Service'ten çıkar: preStop hook'u `webhook.preStopSleepSeconds` (varsayılan 5) kadar bekler, sonra elindeki incelemeleri bitirir.
Deployment `attestkeep-scanner`	1 replica	Trivy taramasını, cosign doğrulamasını ve deny list gerektirdiğinde `attestkeep-binscan` dosya kontrolünü admission yolunun dışında yapar. Admission trafiği almaz. 1.4.0'dan itibaren lisans sunucusuna da bağlanmaz.
StatefulSet `attestkeep-postgresql`	1 replica, 20 GiB, postgres:16.15-alpine	Chart'la gelen veritabanı. `externalDsn` ile kendi PostgreSQL'inizi verirseniz oluşturulmaz.

Ürünün tüm container'ları root olmayan kullanıcıyla (uid 65532), salt okunur root dosya sistemiyle, tüm capability'ler kaldırılmış ve RuntimeDefault seccomp ile çalışır. Varsayılan request ürün workload'u başına 100m CPU ve 256Mi.

Cluster kapsamında `attestkeep.com` grubunda dört CRD (ImageSecurityPolicy, ImageSecurityReport, AttestationRenewal, ImageSecurityUsage) ve bir ValidatingWebhookConfiguration kurulur. Webhook pod CREATE ve UPDATE isteklerine, `kubectl debug`'ın yazdığı `pods/ephemeralcontainers` alt kaynağı dahil, bakar. Timeout 15 saniye, varsayılan `failurePolicy: Ignore`. Mutating webhook yoktur, ürün workload'larınızı değiştirmez.

Dikkat edilecek yetkiler

- ClusterRole node'ları okur (get, list). Operator, lisanstaki node tavanı için Ready node'ları bu yetkiyle sayar. Yalnız sayıyı tutar, node adını tutmaz.
- Secret'lar cluster genelinde yalnız `get` ile okunur, `list` yetkisi yoktur. Operator, adını pod spec'inde gördüğü pull secret'ı okuyabilir. Bir namespace'teki Secret'ları listeleyemez.
- `pods/eviction` yetkisi yalnız `rbac.allowEnforce: true` ya da `rbac.denyList.takedown: true` ile verilir, ikisinin de varsayılanı false. Pod'u durduran runtime policy'si ve `runtimeReconciliation: enforce`, ClusterRole başka ne verirse versin `rbac.allowEnforce: true` ister. Yoksa reddedilir ve kaydedilir. Take down için verilen yetki yalnız deny list içindir.
- Deployment, StatefulSet ve ReplicaSet'lerin scale alt kaynağı (get, update) yalnız `rbac.denyList.takedown: true` ile verilir. Yönetici bununla workload'u sıfıra indirip geri açar. Kaç replica çalışacağını değiştirebilir, ne çalışacağını asla.
- Deployment, ReplicaSet, StatefulSet, DaemonSet, Job ve CronJob'lar 1.4.0'dan itibaren her kurulumda okunur (get, list). Operator pod'un üstündeki sahiplik zincirini okur, böylece bir ret ReplicaSet'e değil Deployment'a sayılır. Yalnız okuma yetkisidir.
- Operator Kubernetes event'i oluşturur: deny list reddinin durdurduğu workload'a ve PEM public key olmayan imza anahtarı taşıyan policy'ye Warning. Kendi policy'lerinin status alanına da yazar, `SignatureKeysValid` koşulu oradadır.
- Namespace kapsamlı bir Role, operator'ün kendi namespace'indeki üç Secret'a yazmasına izin verir: webhook sertifikası, `attestkeep-evidence-key` ve yalnız Falco entegrasyonu açıkken oluşan `<release>-falco-mtls`. 1.4.0'dan itibaren `attestkeep-evidence-key` kurulumun lisans kimliğidir de (12. bölüm).
- İkinci bir Role, `<release>-rule-validation`, operator'ün kendi namespace'inde pod ve ConfigMap oluşturup okuyup silmesine ve pod log'u okumasına izin verir. Falco rule editörü doğrulama pod'unu bununla çalıştırır. Başka namespace'te pod başlatamaz.

Dışarıya giden bağlantılar

Firewall talebiniz için tablo bu. Analitik, hata raporlama ya da telemetri SDK'sı yoktur.

Hedef	Gerekli mi?	Ayar	Kapalıysa
lic.attestkeep.com:443	Evet, offline aktivasyon yoksa	`license.serverUrl`; yolda özel CA ya da TLS'i açan bir proxy varsa `license.caBundle`	Online aktivasyon yapılamaz. Aktive edilmiş kurulum elindeki sertifikayla çalışır, sertifika bitince Community'ye düşer. 1.4.0'dan itibaren günlük kontrolü tek bir operator replica'sı yapar.
ghcr.io (Trivy zafiyet veritabanı):443	Tarama için	`trivy.dbRepository` ile mirror	Mevcut veritabanı çalışmaya devam eder ama eskir. Ne kadar eskidiğini `attestkeep_vulnerability_db_age_seconds` gösterir.
Kendi registry'leriniz:443	Tarama için	Pod spec'indeki pull secret'lar	O registry'deki image'lar taranamaz. Admission, policy'nin taranmamış image ayarına göre davranır.
ghcr.io/attestkeep/attestkeep-k8s:443	Kurulumda	`image.repository` ile mirror	Pod'lar başlayamaz.
Sigstore TUF ve public Rekor:443	Hayır	`scan.cosignOffline: true`	Anahtarlı imza ve offline modla doğrulama cluster içinde kalır.
Sürüm bildirimi	Hayır, varsayılan kapalı	`updates.endpoint`	Adres verilmemiş kurulum hiçbir yere bakmaz.
SIEM hedefleriniz	Hayır, siz tanımlarsınız	Ayarlar > SIEM aktarımı	Olaylar bekler ve yeniden denenir. 24 saat sonra hâlâ ulaşmamış olay bırakılır ve sayfada sayılır.
OTLP collector, Slack, Teams, webhook adresleri, SMTP, SSO sağlayıcı	Hayır, siz tanımlarsınız	Özelliğe göre	O özellik çalışmaz ve log'a yazılır. Bildirim hatası admission'ı durdurmaz.

Lisans sunucumuz ele geçirilirse

Cluster'ınıza komut veremez. Lisans protokolünde iki tür cevap var: sertifika ve yenileme durumu. Sunucunun cevabının bir policy, namespace, image kuralı, URL ya da çalıştırılacak program seçtiği bir kod yolu yoktur. Sertifikalar Ed25519 ile imzalanır, binary'ye gömülü public key ile doğrulanır ve cluster'ın parmak izine bağlıdır.

Kötü niyetli bir sunucu *revoked* cevabı verebilir. Bu durumda kurulum yalnız uyarı veren admission'a geçer, bu da konsolda ve her admission cevabında görünür. Workload yerleştiremez, policy değiştiremez. Sunucu hiç cevap vermezse bir şey olmaz: ağ hataları kesinti sayılır ve kurulum sertifikası bitene kadar onunla çalışır.

3. Admission kararı nasıl verilir

Webhook cevabı cache'ten verir. Tarama admission yolunda çalışmaz, bu yüzden yavaş bir registry deployment'ı yavaşlatmaz.

1. API server bir pod için sorar: pod oluşturulurken ya da güncellenirken, bir de `kubectl debug` çalışan bir pod'a container eklediğinde.
2. Image tag'i digest'e çözülür. Sonuç bir dakika cache'te tutulur. Aynı anda gelen yüz benzer deployment registry'ye tek istek yapar.
3. Digest'in tarama sonucu ve imza sonucu cache'ten okunur.
4. Registry'ye gitmeyi gerektirmeyen kurallar önce bakılır: registry izin listesi, `:latest` ve digest zorunluluğu.
5. Deny list'e bakılır, break-glass'tan ve policy'nin diğer bütün kurallarından önce. Listedeki image, taraması temiz olsa da, break-glass açıkken de reddedilir (7. bölüm).
6. Policy değerlendirilir. Cevap gerekçesiyle birlikte izin ya da rettir.
7. Karar kaydedilir, izinler de dahil. Yalnız retleri listeleyen bir kanıt, kontrolün çalıştığını gösteremez.

Bu adımların hiçbiri taramayı beklemez. Digest ilk kez görülüyorsa üçüncü adımın cevabı yoktur ve ne olacağına policy karar verir.

Falco event'leri bu yolda yer almaz. Bir runtime event'i pod'un kabul edilip edilmeyeceğini değiştirmez.

Dry run. 1.4.0'dan itibaren `kubectl apply --dry-run=server` gerçek bir istek gibi değerlendirilip cevaplanır ama arkasında iz bırakmaz: ledger satırı, deny list eşleşmesi, Kubernetes event'i, bildirim ya da metrik oluşmaz, image kotasına sayılmaz, tarama kuyruğa girmez.

Veritabanı cevap vermediğinde. Bir admission incelemesi veritabanı işine toplam iki saniye ayırır. Sonra statik kurallar hemen karar verir: `:latest`, registry izin listesi ve digest zorunluluğu reddetmeye devam eder. Kayıt ve cache'e dayanan kararlar zayıflar, kesinti log'a bir kez yazılır. 1.4.0'dan önce veritabanına ulaşılamayınca replica Service'ten çıkıyor ve admission cevapları kayboluyordu. Ayrıntısı 11. bölümde.

Debug container'ları. 1.3.7'den itibaren `kubectl debug` ile eklenen container da webhook'tan geçer. Yalnız eklenen container için karar verilir. Image'ı taranır ve policy'ye tabi tutulur, enforce modundaki policy onu diğer image'lar gibi reddeder. Pod'un mevcut container'ları yeniden incelenmez. Debug image'ı aylık image kotasına ve node tavanına sayılmaz, ledger satırındaki gerekçe `; subresource ephemeralcontainers` ile biter. 1.3.7'den önce bu container'lar, image'ı ne olursa olsun, webhook'a hiç gelmiyordu.

Duruşunuzu belirleyen iki ayar

Ayar	Varsayılan	Anlamı
`webhook.failurePolicy` (chart)	`Ignore`	Operator cevap vermezse API server ne yapar. `Ignore` kesinti boyunca pod'ları kontrolsüz kabul eder. `Fail` operator cevap verene kadar reddeder.
`admissionTiming` (policy)	`AllowAndScan`	İlk kez görülen digest'e ne olur. `AllowAndScan` kabul eder ve taramayı kuyruğa koyar. `DenyUntilScanned` tarama kaydı oluşana kadar reddeder. Genellikle `Fail` ile birlikte kullanılır.

İkisinin de herkese uyan doğru değeri yok. `Ignore` operator kesintisinde cluster'a deploy edilebilmesini sağlar, karşılığında kanıtta boşluk kalır ve kanıt paketi bunu yazar. `Fail` boşluksuz bir kontrol sağlar, karşılığında operator kesintisi deployment'ları durdurur. İki ayar da Community dahil her planda var.

Boşluk sırasında içeri giren şey sonrasında da çalışır. Bunu runtime reconciliation taraması bulur. Belirli aralıklarla (varsayılan saatte bir) çalışan her container'ın image ID'sini admission ledger'ıyla karşılaştırır. Arkasında admission kaydı

olmayan digest raporlanır, bildirilir ya da `runtimeReconciliation: enforce` ile durdurulur ve webhook'tan yeniden geçer.

> **Yeni bir digest'in ilk taraması arka planda yapılır.** Yeni kurulmuş bir cluster'da birkaç dakika sürebilir: scanner image'ı çekmek zorundadır ve zafiyet veritabanı henüz hazır değildir. Tarama bitene kadar image Images sayfasında *Taranmamış* durumunda görünür. Aynı digest'in sonraki admission'ları bitmiş rapora göre karar verir. *Yeniden tara* istendiğinde tarama başlatır.

4. Kurulum, yükseltme ve ilk açılış

Gerekenler

- Kubernetes 1.27 ya da üstü.
- ValidatingWebhookConfiguration oluşturma yetkisi. İlk kurulumda cluster-admin gerekir.
- Chart'la gelen PostgreSQL için yaklaşık 20 GiB, ya da kullandığınız bir PostgreSQL'in DSN'i. SIEM aktarımı PostgreSQL 13 ya da üstünü ister, chart'la gelen 16'dır.
- attestkeep.com'da bir hesap ve lisans anahtarı: ücretsiz Community anahtarı ya da satın aldığınız plan.

Kurmadan önce release'i doğrulayın

Her release digest'i üzerinden imzalanır. Anahtarlı kontrol için registry'niz ve docs.attestkeep.com/cosign.pub adresindeki public key yeterli.

```
curl -sO https://docs.attestkeep.com/cosign.pub
cosign verify --key cosign.pub ghcr.io/attestkeep/attestkeep-k8s:1.4.0
```

Kurulum

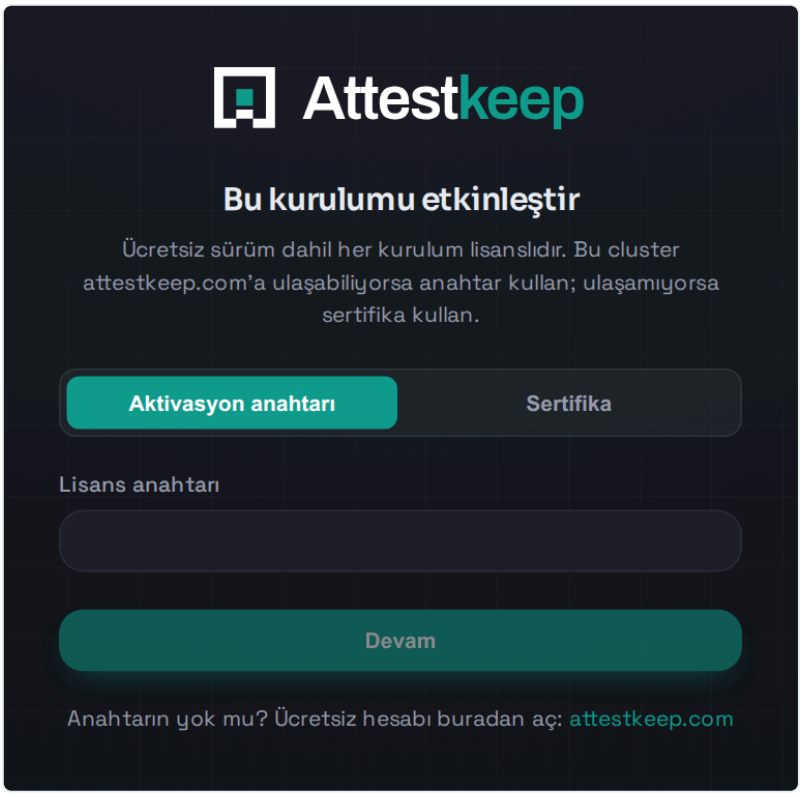
```
# doğrudan registry'den kurar; `helm repo add` gerekmez
helm install attestkeep oci://ghcr.io/attestkeep/charts/attestkeep \
  --version 1.4.0 \
  --namespace attestkeep --create-namespace \
  --set clusterName=production-eu \
  --wait
```

`clusterName` değerini deneme kurulumunda bile ilk seferde verin. Bildirimler, kanıt belgeleri, panel ve Ayarlar > Hakkında kurulumu bu adla anar. Değer verilmezse ilk node'un adı kullanılır. Adı siz seçersiniz, cluster'dan hostname ya da adres çıkmaz.

Çalıştığını kontrol edin ve konsolu açın

```
kubectl -n attestkeep get pods
kubectl -n attestkeep port-forward svc/attestkeep 8080:8080
```

`http://localhost:8080` adresini açın. Yeni kurulumda iki ekran bir kez gelir: aktivasyon ve owner hesabı. Sonra konsol güvenlik özetiyle açılır (5. bölüm).



Ekran 1, *Bu kurulumu etkinleştir*. Ücretsiz sürüm dahil her kurulum lisanslıdır. *Aktivasyon anahtarı* sekmesinde anahtarınızı yapıştırıp devam edersiniz. Anahtar, operatörün cluster içinde doğruladığı imzalı bir sertifikayla değiştirilir. attestkeep.com'a erişemeyen bir cluster bunun yerine *Sertifika* sekmesini kullanır (12. bölüm).



Ekran 2, *Hesabını oluştur*. *Kayıt ol* düğmesinin altındaki satır lisansın etkin olduğunu ve bu hesabın kurulumun sahibi olacağını söyler. Hesap oluşunca ekran kapanır. Community anahtarıyla kurduktan sonra bir plan aldıysanız ücretli anahtarı daha sonra Kullanım ve Lisans sayfasında etkinleştirirsiniz.

Aktivasyon lisans sunucusuna ulaşamazsa ekran bunu nedeniyle söyler. Operator'ün güvenmediği bir sertifika (`license.caBundle` verilmemiş özel bir CA), reddedilen ya da zaman aşımına uğrayan bağlantı ve çözülemeyen ad aynı

cümleyle gelir: "Lisans sunucusuna ulaşılamadı: sertifikasına güvenilmiyor ya da bağlantı kurulamadı. Sunucu özel bir CA kullanıyorsa chart'ta license.caBundle ayarlayın." 1.4.0'dan önce bu hatalar geçersiz anahtar gibi görünüyordu. Mesajların listesi 12. bölümde.

Kendi veritabanınızı kullanmak

Migration'lar binary'nin içindedir ve ilk bağlantıda uygulanır. Her biri kaydedilir, bir kez çalışır. Kullandığınız bir veritabanını vermek için DSN'i komut satırına değil Secret'a koyun:

```
kubectl -n attestkeep create secret generic attestkeep-db \
  --from-literal=dsn='postgres://user:pass@db.internal:5432/attestkeep?sslmode=require'

helm upgrade attestkeep oci://ghcr.io/attestkeep/charts/attestkeep \
  --namespace attestkeep \
  --set postgresql.enabled=false \
  --set externalDsnExistingSecret.name=attestkeep-db
```

Private registry için ayar gerekmez. Operator, pod'un kullandığı pull secret'larla ve service account'taki secret'larla hem taramayı hem imza kontrolünü yapar.

Kaldırma `helm uninstall` ile yapılır. Webhook yapılandırması release'le birlikte silinir ve admission denetimi o anda durur. Veritabanı volume'u silinmez.

Yükseltme

Yükseltme `helm upgrade` ile yapılır. Gömülü migration'lar yeni olanları uygular. `helm upgrade --reuse-values` bu sürümler arasında çalışır, kurulu sürümden sonra eklenen değerler chart varsayılanını alır. Migration'lar yalnız ileri gider. Aşağıda migration'ı olan bir adımda eski sürüme dönmek `helm rollback` ile değil, veritabanı yedeğinden geri yüklemeyle olur.

Geçiş	Migration	Yapılacak
1.3.7 → 1.4.0	0036–0046	Bu yükseltmeyi `--set webhook.failurePolicy=Fail` ile yapın ve sonra eski değere döndürün, ya da değişiklik dondurma döneminde yapın. 1.3.7 replica'sı boşaltmadan kapanır ve yaklaşık bir saniye boyunca ona gelen istek `Ignore` altında kontrolsüz kabul edilir. 1.4.0'dan sonraki her yükseltme boşaltarak yapılır. Her migration kilidi için en fazla beş saniye bekler, sonra replica yeniden başlayıp tekrar dener. 1.3.7'ye dönmek yedekten geri yüklemeyle olur. Yeni chart değerlerinin hepsi isteğe bağlıdır, `rbac.denyList.takedown: true` verilene kadar take down kapalı kalır. ClusterRole pod'un üstündeki workload nesnelerini okuma yetkisi kazanır. Mevcut bildirim kanalları *Deny list eşleşmesi*'ne abone olmaz, retlerin bildirilmesini istediğiniz kanalda işaretleyin. Harici veritabanı SIEM aktarımı için PostgreSQL 13 ya da üstü olmalı. 1.4.0'dan önce offline aktive edilmiş kurulumda, imzalı kanıt indirmesi için yeni bir aktivasyon isteği gerekir (12. bölüm).
1.3.6 → 1.3.7	0033–0035	Yalnız image'ı değiştirmek yetmez, `helm upgrade` ile yükseltin: `pods/ephemeralcontainers` için yeni webhook kuralı chart'ın içinde. Bu sürümden itibaren policy'nin izin vermediği bir image ile `kubectl debug` reddedilir. 0033 What-if koşularını saklayan tabloyu ekler, 0034 images tablosuna iki sütun ekler, 0035 bunları ledger'dan doldurur. 1.3.6'ya dönmek yedekten geri yüklemeyle olur. Yeni chart değeri yok.
1.3.2 → 1.3.6	Yok	Bir şey gerekmez. 1.3.3, 1.3.4, 1.3.5 ve 1.3.6 migration ya da yeni chart değeri getirmez.
1.3.1 → 1.3.2	Yok	Operator için bir şey gerekmez, 1.3.1'e `helm rollback` çalışır. 1.3.1 rule editöründeki falcoctl değerlerini kopyaladıysanız `install.refs` ve `follow.refs` listelerine `falco-rules:5`'i geri ekleyin. Eklenmezse falcoctl Falco'nun kendi `/etc/falco/falco_rules.yaml` dosyasını kurmaz.
1.3.0 → 1.3.1	0026–0032	Policy kaynağında artık `spec.runtime` alanı yok. Policy dosyalarınızda varsa çıkarın: `kubectl apply` bu alanı taşıyan policy'yi `unknown field "spec.runtime"` hatasıyla reddeder. 0031 migration'ı, kaldırılan aksiyonla kaydedilmiş runtime kurallarını issue açan notify kuralına çevirir. Yeni bildirim kanalları Runtime uyarısına abone açılır, mevcut kanalların abonelikleri değişmez.
1.2.x → 1.3.0	0025	Cluster'ı yeniden aktive edin. 1.3.0 ve sonrası node tavanını taşıyan sertifika ister. Eski sertifikayı "re-activate with a key issued by licence service 1.3.0 or later" mesajıyla reddeder.
1.1.x → 1.2.0	0021–0024	Falcosidekick üzerinden bearer token ile gelen Falco event'leri, Falco mutual TLS ile bağlanana kadar gelmez (8. bölüm). Kullanıcı silmek artık hesabı kapatır.

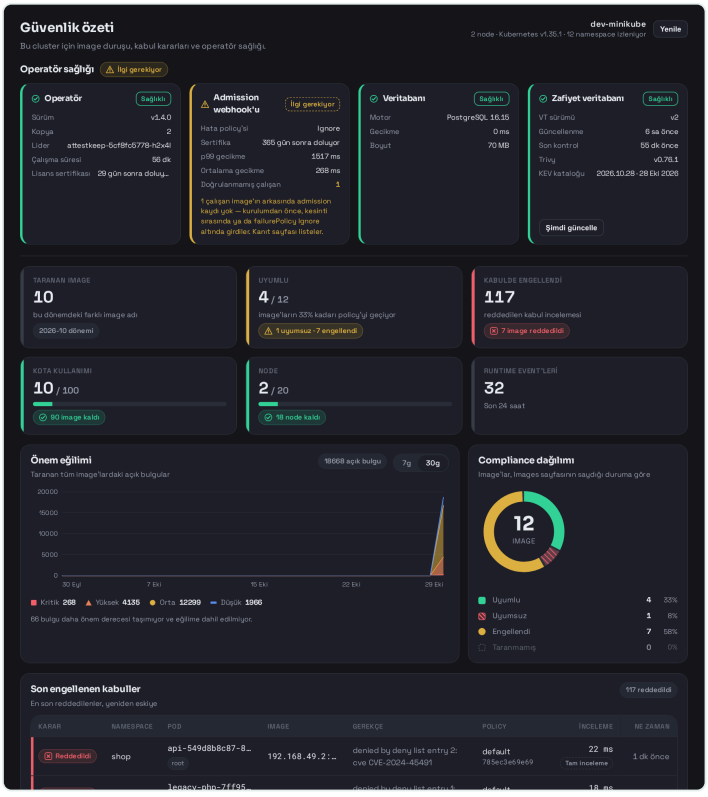
RBAC'ini kendisi yöneten kurulum, Falco entegrasyonunu açmadan önce `<release>—falco-mtls` Secret'ını Role'e eklemelidir.

5. Konsol

Konsol release'in parçasıdır, operator'ü kuran Helm release onu da sunar. Dışarıdan font, script ya da analitik yüklemez. Türkçe ve İngilizce, açık ve koyu tema seçenekleri var, her ekranda yardım paneli bulunur. Dil ve tema Ayarlar > Görünüm'den değişir.

Kenar menüde dört grup var. *Güvenlik*: Images, Vulnerabilities, Policies, Deny list, What-if geçmişi. *Falco*: Policy'ler, Rule'lar, Event'ler, Falco entegrasyonu. Bu grup Falco entegrasyonu açıkken görünür. *Compliance*: Compliance kanıtı ve Denetim Kaydı. *Ayarlar*: Şirket, Görünüm, Bildirimlerim, Webhooklar, Tarama, SMTP, Çoklu Oturum Açma, Günlükleme ve telemetri, SIEM aktarımı, Kullanım ve Lisans, Kullanıcılar, Yedekleme ve Hakkında. 1.4.0'da Türkçe konsol image, policy, node, admission, framework, compliance ve Vulnerabilities gibi terimleri ve plan adlarını İngilizce bırakır. Bu kitap menü ve düğme adlarını konsolda göründükleri gibi yazar.

Güvenlik özeti



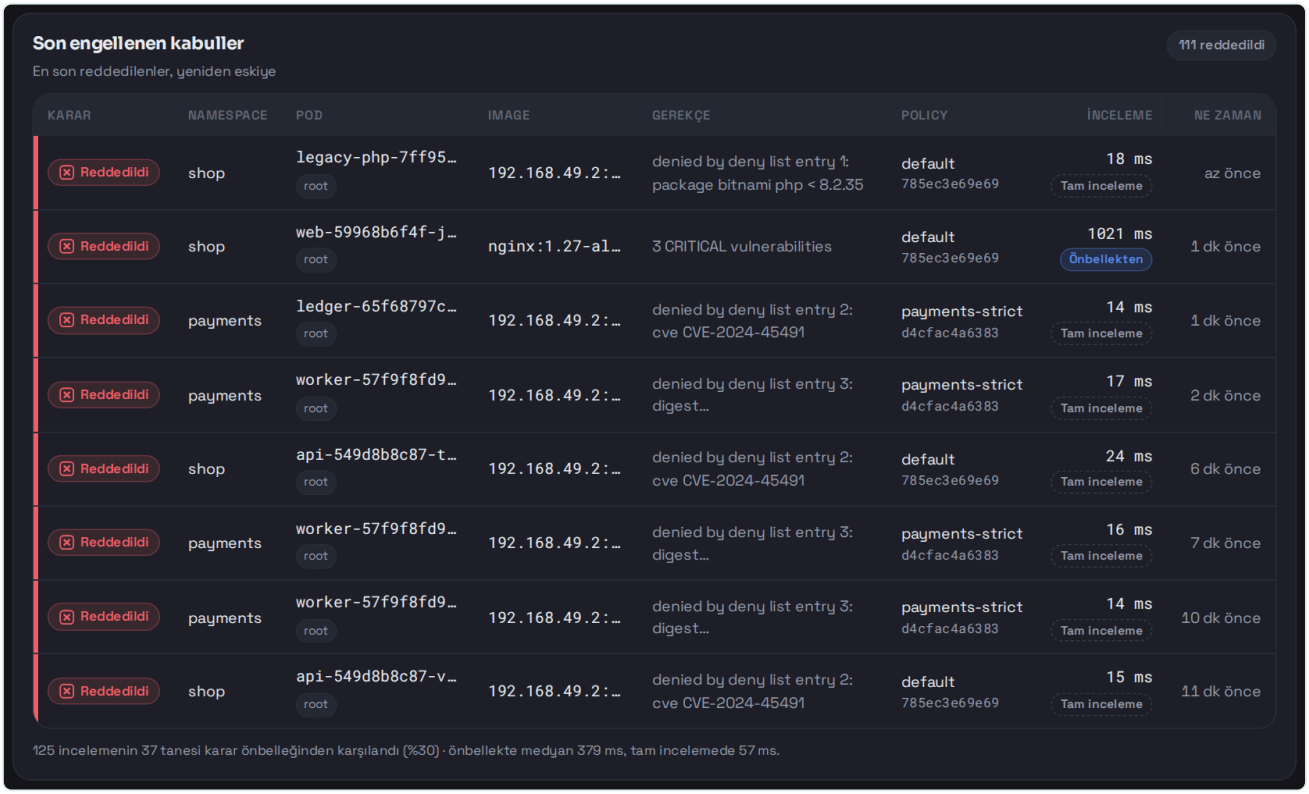
Bu kitap için fotoğraflanan kurulumun güvenlik özeti: iki kopyayla 1.4.0, iki node, Kubernetes v1.35.1. *Admission webhook'u* kutusu ilgi istiyor, çünkü çalışan bir image'ın arkasında admission kaydı yok. Kutu bunun nasıl olabileceğini de yazıyor. Altta: bu dönem taranan 10 image adı, 12 image'dan 4'ü uyumlu, 1'i uyumsuz, 7'si engellendi, reddedilen 117 admission incelemesi, *Kota kullanımı* 100'de 10, *Node* 20'de 2 ve son 24 saatte 32 runtime event'i. Compliance dağılımı Images sayfasıyla aynı dört durumu sayar.

1.3.5'ten itibaren bu iki kart kendiliğinden uyarı vermez. Webhook kartı gecikmeyi sabit bir sınırla karşılaştırmaz, çünkü bir image'ın ilk incelemesi digest'i registry'den çözer ve cache'ten cevap veren sonraki incelemelerden uzun sürer. Operator zafiyet veritabanını ve KEV kataloğunu altı saatte bir kendisi yeniler. Yönetici *Şimdi güncelle*'ye basarak indirmeyi hemen başlatabilir. Kart sonra veritabanının değiştiğini, zaten güncel olduğunu ya da indirmenin neden başarısız olduğunu yazar. İstek audit log'a `vulnerability_db_refresh` olarak geçer.

Node kartı 1.3.1 ile geldi. Cluster tavanın üstündeyken şerit ne zamandan beri aşıldığını ve yeni image adlarının hangi tarihten itibaren reddedileceğini gösterir. O tarih geçince enforcement'ın başladığını yazar. İki durumda da çalışan hiçbir şey durmaz.

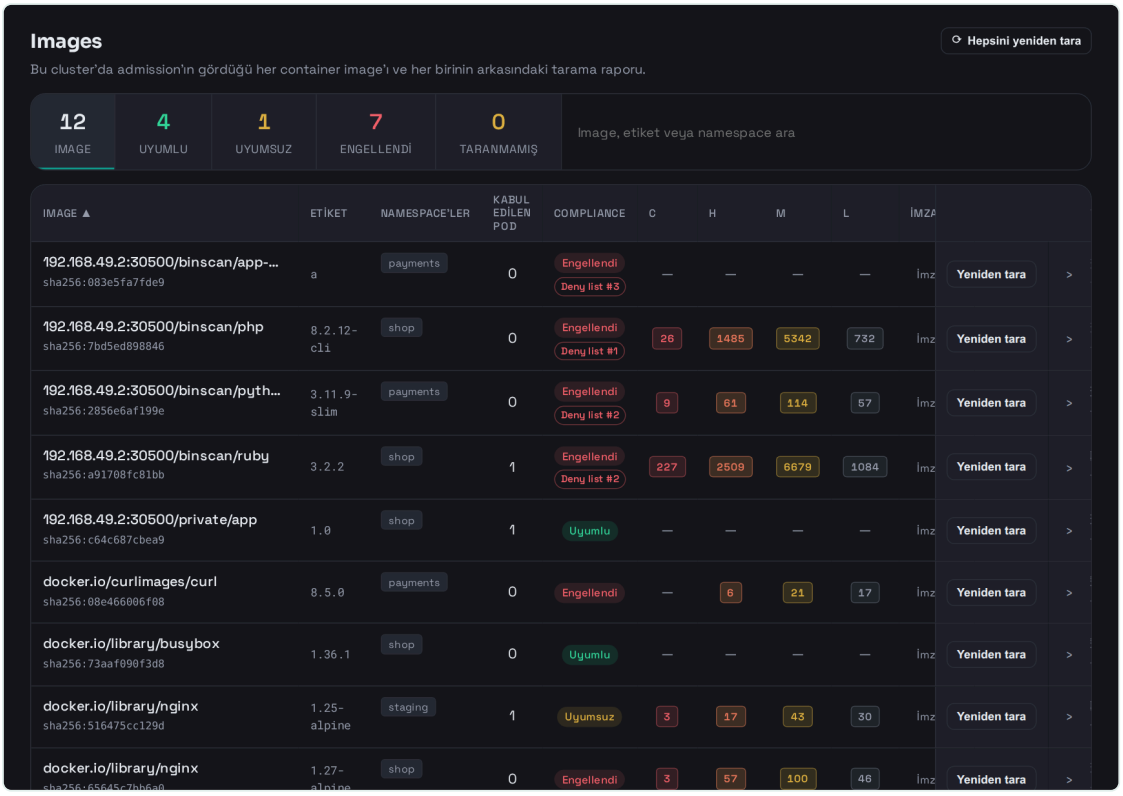
1.4.0'dan itibaren compliance kartı, çipleri ve dağılım grafiği Images sayfasıyla aynı dört durumu aynı kuralla sayar, iki sayfa aynı rakamı verir. Son admission'ı reddedilen image, taraması policy'ye uysa bile Engellendi'dir. Break-glass ile kabul edilen

image Uyumsuz sayılır. Dağılımdaki Acil erişim ve Bilinmiyor dilimleri kalktı. *Önem eğilimi* tutulan her image'ın açık bulgularını, yani Vulnerabilities sayfasının satırlarını sayar, başlığı da altındaki dört bandın toplamıdır. *Lider*, controller kilidini tutan replica'yı gösterir.



Son engellenen kabuller yalnız retleri, en yenisi üstte listeler. Deny list'in verdiği ret kaydı adıyla anar, örneğin *denied by deny list entry 1: package bitnami php < 8.2.35*. Policy'nin kendi kurallarından gelen ret kuralı söyler, burada 3 CRITICAL zafiyet. Her satırda policy ve hash'i, inceleme süresi ve kararın önbellekten gelip gelmediği var. Tablonun altındaki satır önbellekten ve tam incelemeyle verilen kararları karşılaştırır.

Images



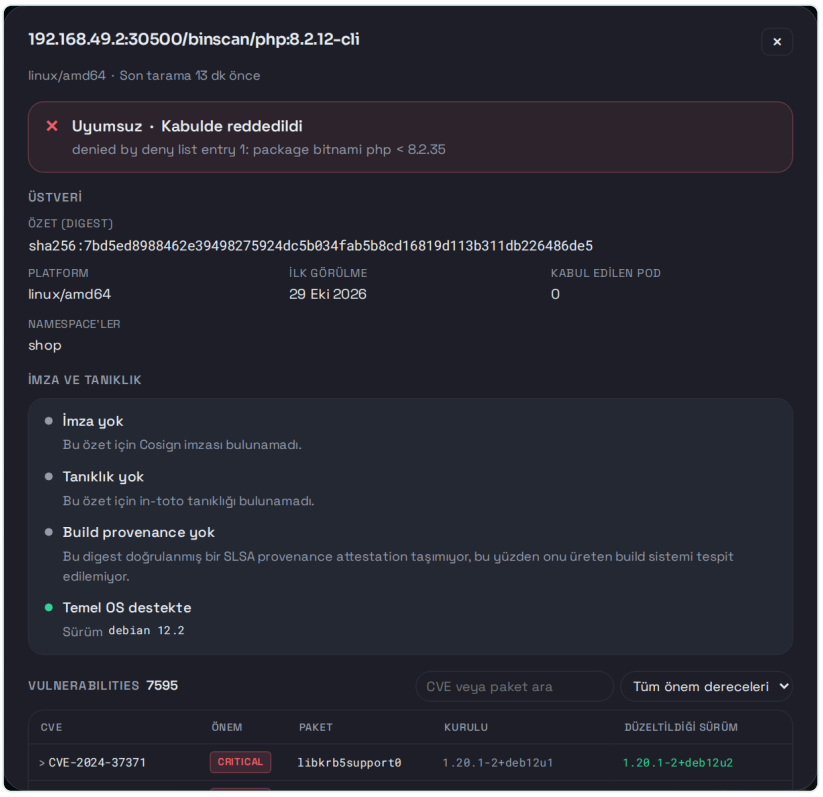
Admission'ın bu cluster'da gördüğü her image: 12 image, 4 uyumlu, 1 uyumsuz, 7 engellendi, taranmamış yok. Kutuların toplamı listeye eşit. Deny list'in reddettiği satır, durumunun altında *Deny list #N* rozetini taşır. Sütunlar namespace'leri, kabul edilen pod'ları, önem derecesine göre bulguları ve imza durumunu verir. *Yeniden tara* tek image'ı, *Hepsini yeniden tara* bütün image'ları kuyruğa alır.

1.3.7'den itibaren her image dört durumdan yalnız birindedir, bu yüzden kartların toplamı listeye eşittir. Uyan ilk durum geçerlidir:

- **Engellendi**: son admission incelemesi reddedildi.
- **Taranmamış**: kabul edildi, henüz tarama kaydı yok.
- **Uyumlu**: kabul edildi, tarandı, policy'ye uyuyor.
- **Uyumsuz**: kabul edildi ve tarandı ama policy'ye uymuyor. Genellikle audit modundaki bir policy ya da break-glass penceresi altında olur.

Engellendi durumu, scanner'ın sonradan vardığı sonuca değil, son admission incelemesine bakar. Audit modundaki bir policy altında ilk taramadan önce kabul edilen, sonra kritik bulgusu çıkan image Uyumsuz'dur ve detayında "Admitted; the next admission would be denied" yazar. 1.4.0'dan itibaren her satırın Compliance çipi durumunu aynı sözcüklerle yazar, filtreler de buna göre seçer. Taramanın kendi sonucu image detayındadır. Deny list'in reddettiği satırda *Deny list #N* rozeti durur, üzerine gelince ret görünür. *Kabul edilen pod* sütunu image'la, ilk görüldüğünden beri kabul edilmiş pod'ları sayar. Pod durunca azalmaz, reddedilen admission hiç sayılmaz.

1.3.1'den itibaren Images ve Vulnerabilities listeleri sayfa sayfa okunur. Filtre, arama ve sıralama operator'de çalışır. Tablonun üstündeki sayılar tüm satırları kapsar.

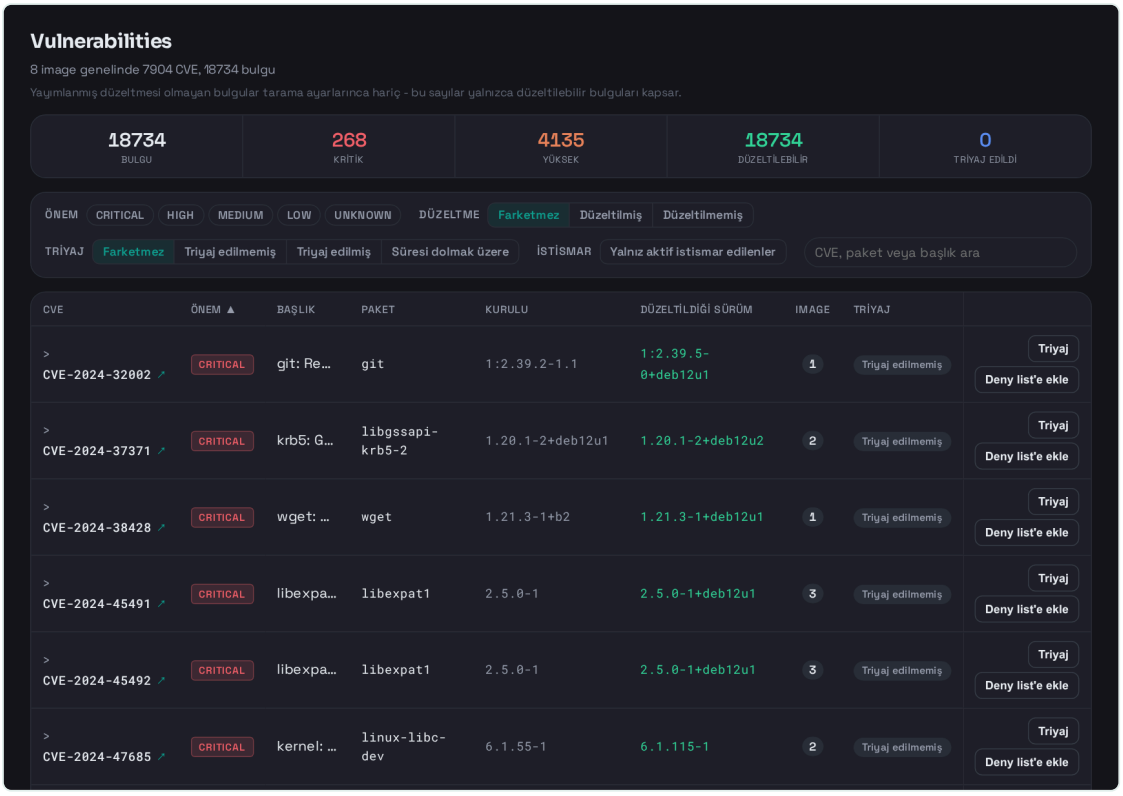


`php:8.2.12-cli` paneli. Admission bu image'ı deny list kaydı 1 yüzünden reddetti ve panel reddi aynen aktarır. Bulguların üstünde dört soru cevaplanır: cosign imzası var mı, in-toto attestation var mı, doğrulanmış SLSA build provenance var mı, temel OS hâlâ destekte mi. Burada temel OS Debian 12.2 destekte, tarama 7595 zafiyet bildiriyor.

Detayda image'ın digest'iyle dolu *Deny list'e ekle* düğmesi bulunur. Digest zaten listedeyse *Listede* yazar. Deny list'te dosya kontrolünün cevaplayacağı bir kayıt varsa, detay bu digest için kontrolün cevaplarını da *Deny list kontrolleri* altında gösterir (7. bölüm).

Aynı tarama sonucu kubectl ile de okunabilir. Her `ImageSecurityReport` özeti `status.summary`'de, bulguları `status.vulnerabilities`'te (en ağırı başta, en fazla 1000) ve tarama zamanını `status.lastScanned`'de taşır. Triage kararı da bu nesneye `triagedBy: user:<hesap id>` ile yazılır, isim yazılmaz.

Vulnerabilities ve triage

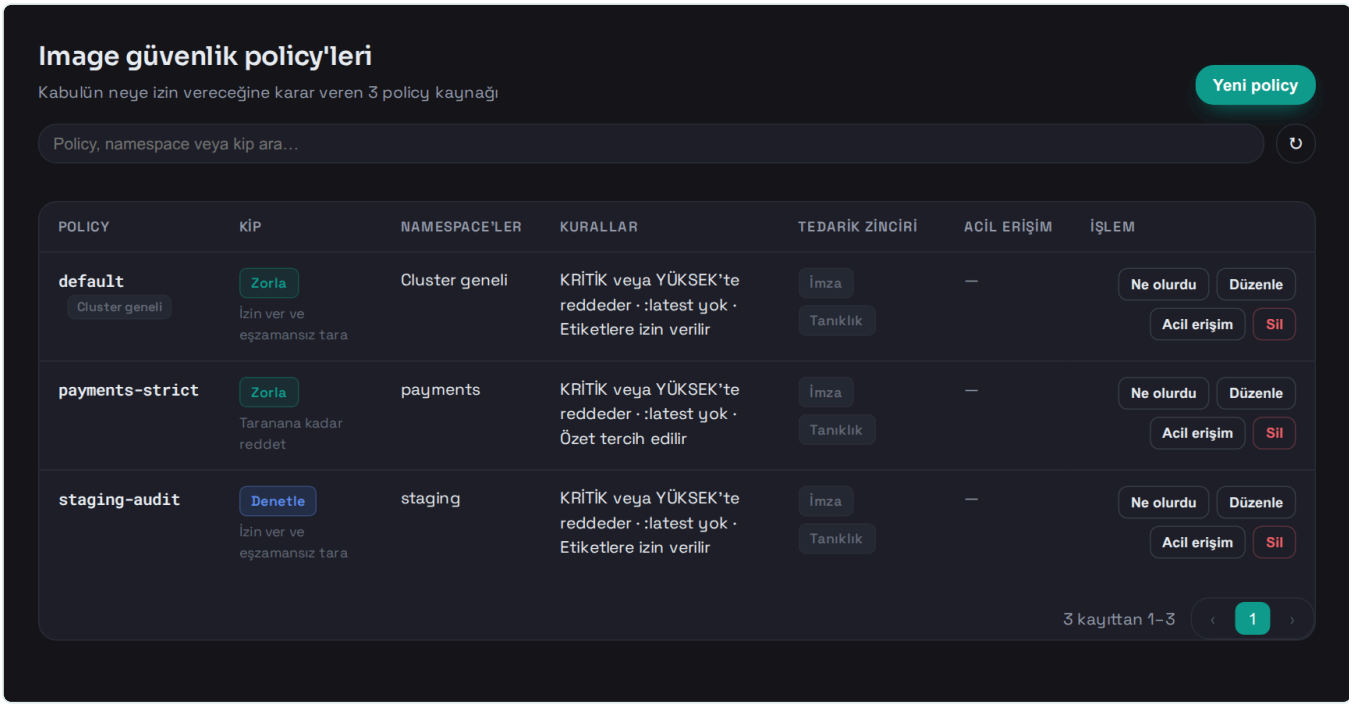


Bütün image'lardaki bulgular CVE'ye göre gruplanmış, kurulu ve düzeltilmiş sürümler yan yana: 8 image'da 7904 CVE ve 18734 bulgu. Başlığın altındaki satır, bu kurulumun tarama ayarlarının düzeltmesi yayımlanmamış bulguları dışarıda bıraktığını söyler. Her satırda *Triyaj* ve 1.4.0'dan itibaren *Deny list'e ekle* var.

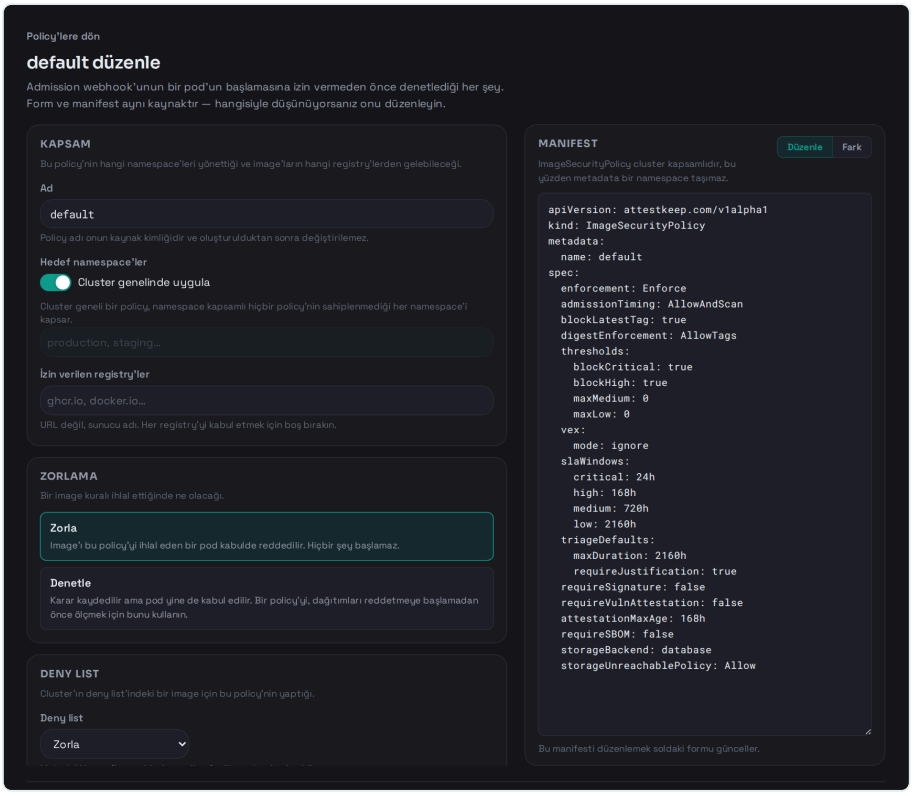
1.4.0'dan itibaren başlık sayfadaki 25'i değil, filtrelerin seçtiği bütün CVE'leri sayar. Yanındaki bulgu ve image sayıları da aynı filtrelerle hesaplanır. İki referansla, tag'iyle ve digest'iyle çekilen bir digest tek image'dır: bulguları bir kez sayılır, her referansıyla birlikte bir kez listelenir. Satırdaki *Deny list'e ekle* advisory id'yi, paket kaydı için de paketi ve kurulu sürümü önerir.

Triage kararı denetçinin okuduğu bir kayıttır, sessiz bir susturma değildir. Durumu seçersiniz (false positive, kabul edilmiş risk ya da inceleniyor), gerekçe yazarsınız, bitiş tarihi koyarsınız. Süre dolunca bulgu yeniden policy sonucuna bağlanır. Her kararda kimin ne zaman verdiği yazılır, karar audit log'a ve sonraki kanıt paketine girer.

Admission policy'leri



Üç policy. `default` cluster genelinde zorlar, `payments-strict` `payments`'ta zorlar ve taranana kadar reddeder, `staging-audit` `staging`'de Denetle modunda çalışır. Her satır modu, kapsanan namespace'leri, kural özetini ve istenen tedarik zinciri kontrollerini gösterir. Satır sonunda *Ne olurdu*, *Düzenle*, *Acil erişim* ve *Sil* var.



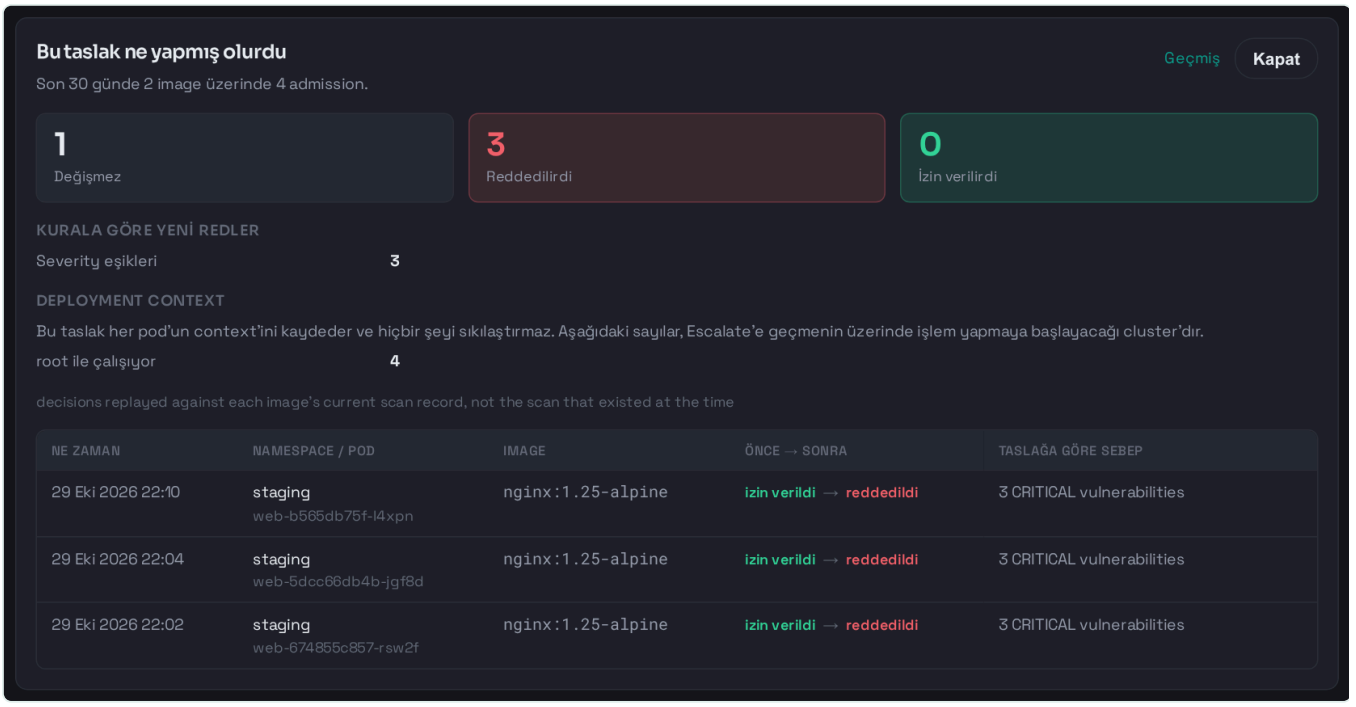
`default` editörü. Form ve manifest aynı kaynaktır, hangisini isterseniz onu düzenlersiniz. Yeni *Deny list* bölümü *Zorla*'ya ayarlı: listedeki image'lar policy Denetle modunda olsa bile reddedilir.

1.3.7'den itibaren *Cluster'a uygula* yalnız policy yazabilen hesaplara görünür. Diğerleri What if sonucunu, İndir'i ve Git için kopyala'yı kullanmaya devam eder. *İptal*, kaydedilmemiş değişiklikleri atmadan önce sorar. 1.4.0'dan itibaren kaydetme, kaynağın formun düzenlemediği kısımlarını korur: kapanmış bir break-glass penceresinin kaydı ya da kubectl ile yazılmış bir blok gibi. Formda *Zorla*, *Denetle* ve *Kapalı* seçenekli bir *Deny list* bölümü vardır (7. bölüm). Yeni policy, operator'ün ve

dokümantasyonun kullandığı SLA ve triage süreleriyle açılır (critical 24h, high 168h, medium 720h, low 2160h, triage 2160h). kubectl ile yazılmış ve bir süreyi boş bırakan policy bu değerlerle açılır, uygulamadan önce diff bunu gösterir.

Bir policy'nin sizin cluster'ınızda neyi reddedeceği yalnız metnine değil, image'larınıza, tarama sonuçlarına ve triage kararlarına bağlıdır. What if (konsolda *Ne olurdu*) bunu cluster'ın kendi kayıtlarından hesaplar. Taslak, son 7, 30 ya da 90 günün admission'ları üzerinde, webhook'un kullandığı fonksiyonla yeniden çalıştırılır. Hiçbir şey uygulanmaz: hiçbir pod'a dokunulmaz, registry'ye gidilmez, lisans sayacı değişmez. Kararlar image'ların o günkü değil bugünkü tarama sonuçlarıyla yeniden hesaplanır.

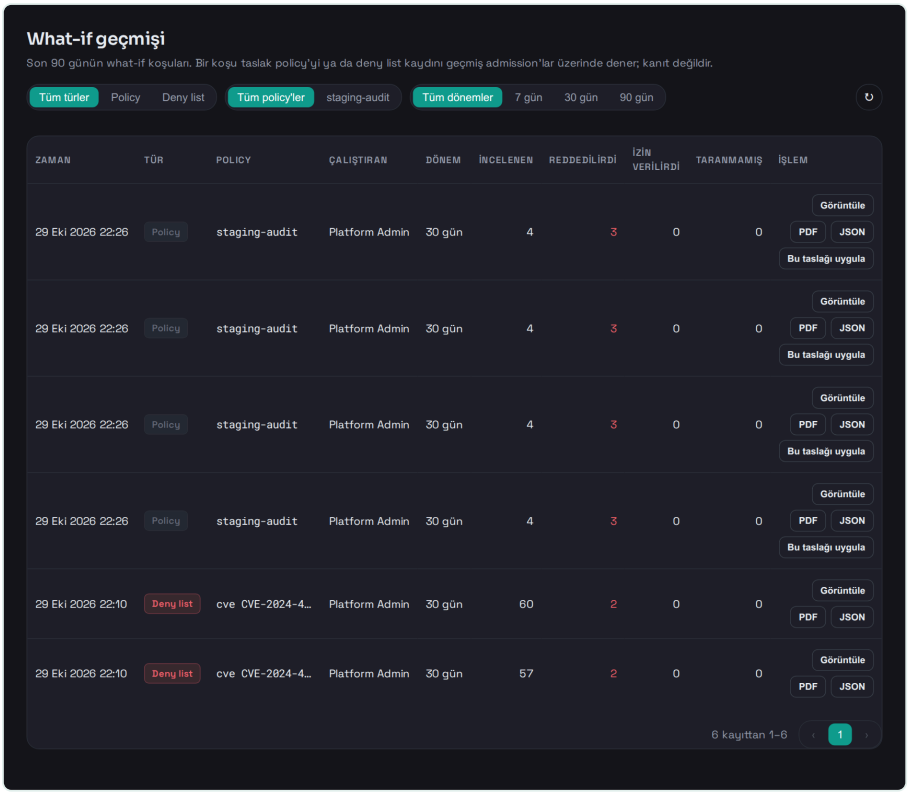
Policy listesindeki *Ne olurdu* kısa yoldur. Audit modundaki policy enforce ediyormuş gibi, enforce modundaki policy olduğu gibi, son otuz gün üzerinde yeniden çalıştırılır. Sonuç editörde açılır, hiçbir şey uygulanmaz.



`staging-audit`, zorlayan bir taslak olarak son 30 güne karşı yeniden oynatıldı: 2 image'da 4 admission, 1 değişmez, 3 önem eşikleri yüzünden reddedilirdi, 0 izin verilirdi. Panel yeni retlerin arkasındaki kuralları, yeniden oynatılan pod'ların deployment context sinyallerini (4'ü root olarak çalışıyor) ve taslağın gerekçesiyle her değişen admission'ı listeler.

Şimdi tara. Dönemde tarama kaydı olmayan image varsa sonuç onları sayar. Deployer ya da yönetici tam olarak bu digest'ler için tarama kuyruğa alabilir, bir istekte en fazla 500. Her istek audit log'a `rescan` olarak geçer. Koşu kendiliğinden yeniden çalışmaz, taramalar bitince *Ne olurdu*'ya yeniden basın.

What-if geçmişi



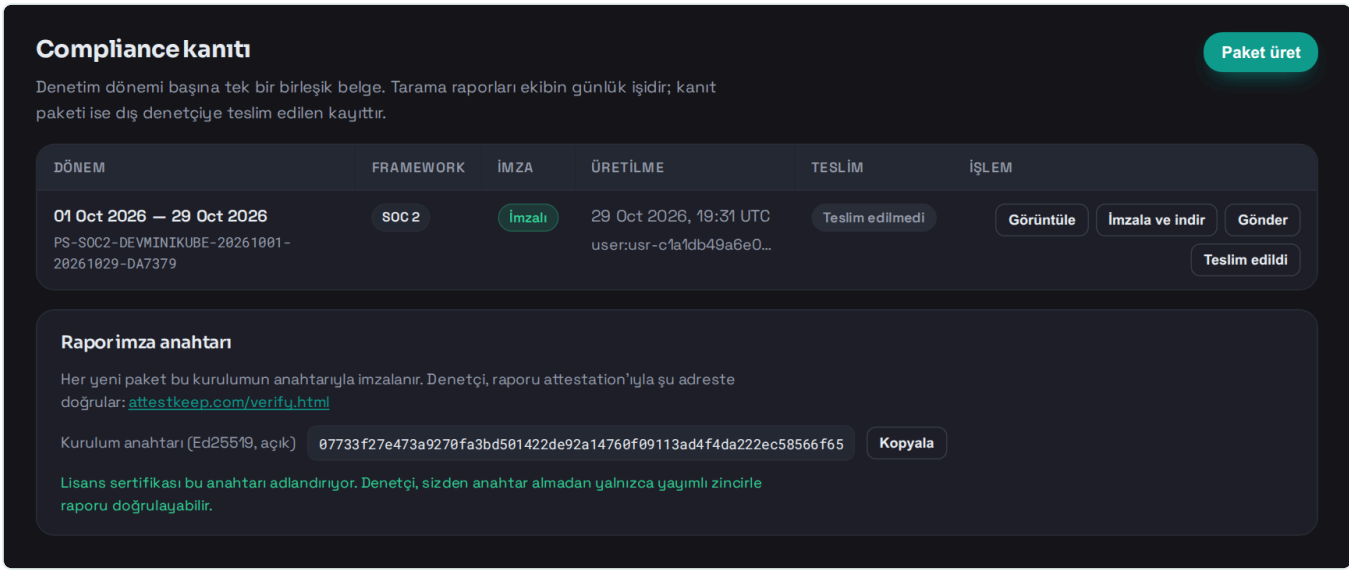
Güvenlik > What-if geçmişi iki tür koşuyu birlikte tutar. *Tür* sütunu ve filtresi policy taslağını deny list kaydından ayırır: burada `staging-audit`'in dört koşusu ve CVE-2024-45491 kaydının iki koşusu var. Her koşu kimin çalıştırdığını, dönemi ve sayıları gösterir. *Bu taslağı uygula* yalnız policy koşularında çıkar.

1.4.0'dan itibaren geçmişte iki tür koşu tutulur: policy taslağı ve eklenmeden önce ya da satırından denenen deny list kaydı (7. bölüm). *Tür* sütunu ve tür filtresi ikisini ayırır. Deployer ya da yöneticinin çalıştırdığı koşu doksan gün, her türden en fazla beş yüz koşu saklanır. Viewer da yeniden çalıştırabilir, cevap gösterilir ama saklanmaz. Her koşu yeniden açılabilir, PDF ya da JSON olarak indirilebilir, yönetici tarafından *Bu taslağı uygula* ile editörde açılabilir. Bu düğme saklanan taslağı yükler, hiçbir şey uygulamaz.

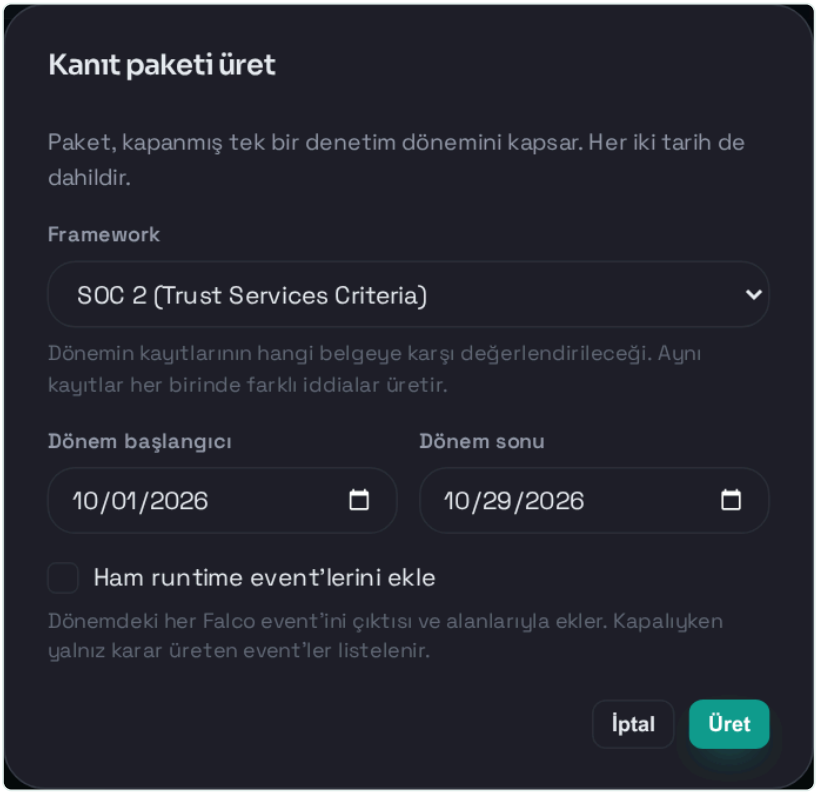
> **İndirilen dosya kanıt değil, projeksiyondur.** PDF bunu her sayfada yazar, imzalı ya da mühürlü değildir. Koşu hiçbir zaman kanıt paketine girmez, çünkü paket ne olduğunu, koşu ne olabileceğini kaydeder. Her indirme audit log'a `policy_whatif_download` olarak geçer.

API `POST /api/v1/policies/whatif`. İsteğe bağlı `source` alır: taslağın düzenlendiği policy (bilinmeyen ad `400` ile cevaplanır). Cevap sayıların yanında `kept`, `run_id`, `unscanned`, `unscanned_total` ve `notes` taşır. Bir hesap her operator replica'sında dakikada en fazla on koşu başlatabilir, sonraki `429` ve `Retry-After` ile cevaplanır.

Compliance kanıtı

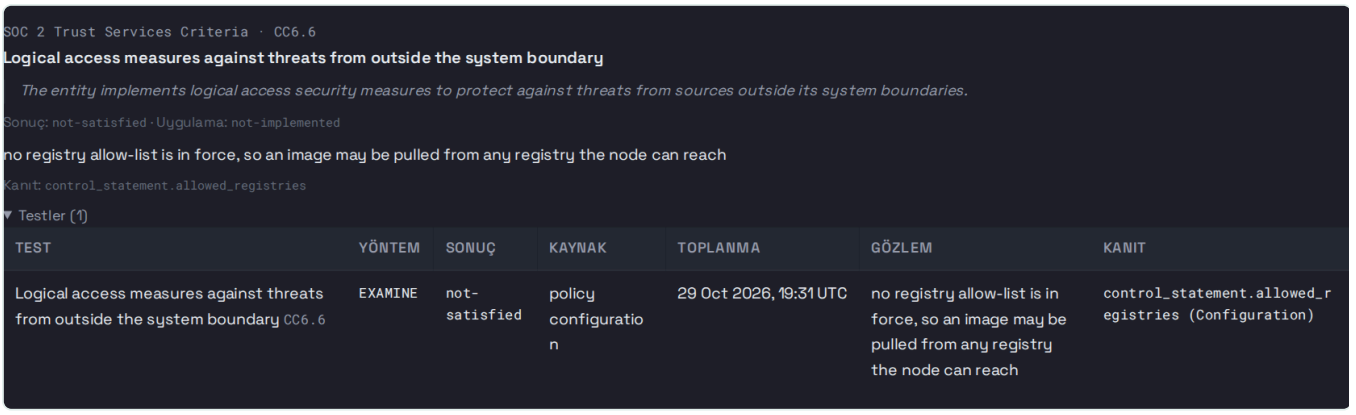


Her denetim dönemi için tek belge. Satır dönemi, framework'ü, imza durumunu, paketin ne zaman ve hangi hesapla üretildiğini (`user:<hesap kimliği>` olarak) ve teslim edilip edilmediğini gösterir. Altında, denetçinin raporu karşılaştırdığı kurulumun açık imza anahtarı var.



Kanıt paketi üret. *Framework* listesinde lisansın izin verdiği eşlenmiş kontrol setleri var. Dönem kapanmış tek bir denetim dönemidir, iki tarih de dahil. *Ham runtime event'lerini ekle* dönemin bütün Falco event'lerini ekler. Kapalıyken yalnız bir karara yol açan event'ler listelenir.

Lisans, kurulumun daha önce kullandığından az framework kapsıyorsa, örneğin ücretli lisans bittiğinde, giriş yapan ilk owner ya da yönetici hangilerinin kalacağını seçer. Ayrıntısı 12. bölümde.



SOC 2 Trust Services Criteria · CC6.6

Logical access measures against threats from outside the system boundary

The entity implements logical access security measures to protect against threats from sources outside its system boundaries.

Sonuç: not-satisfied · Uygulama: not-implemented

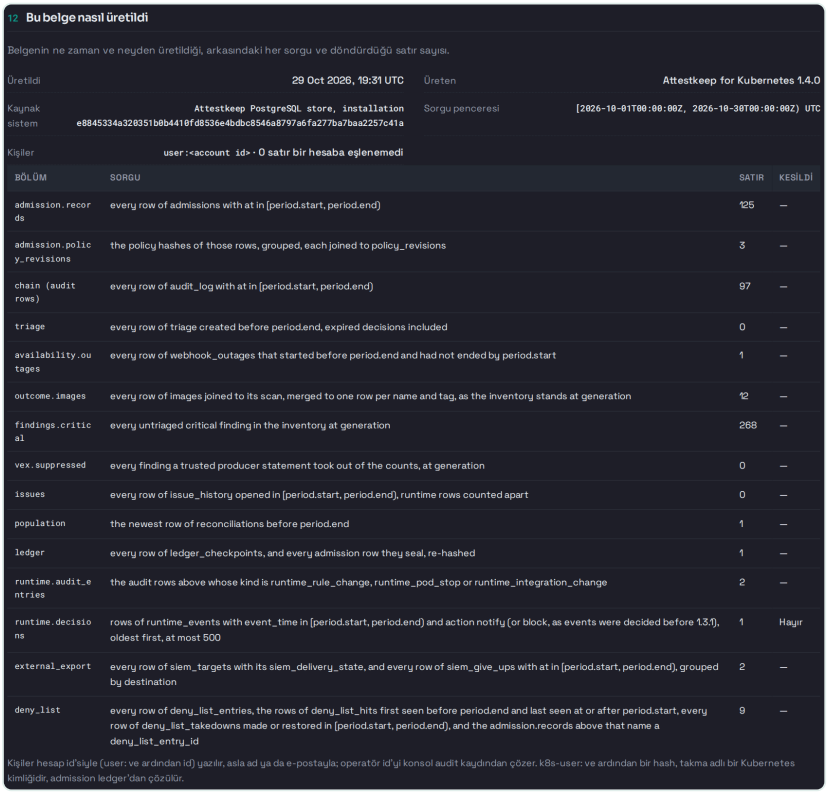
no registry allow-list is in force, so an image may be pulled from any registry the node can reach

Kanıt: control_statement.allowed_registries

▼ Testler (1)

TEST	YÖNTEM	SONUÇ	KAYNAK	TOPLANMA	GÖZLEM	KANIT
Logical access measures against threats from outside the system boundary CC6.6	EXAMINE	not-satisfied	policy configuration	29 Oct 2026, 19:31 UTC	no registry allow-list is in force, so an image may be pulled from any registry the node can reach	control_statement.allowed_registries (Configuration)

Rapor görünümünde testi açılmış bir gereksinim. CC6.6 karşılanmıyor, çünkü yürürlükte bir registry izin listesi yok ve test bunu söylüyor. Yöntem EXAMINE, çünkü bir yapılandırmayı okuyor. Tablo ayrıca sonucu, kaynağı, kaynağın ne zaman okunduğunu, gözlemi ve rapordaki yolu verir.



12 **Bu belge nasıl üretildi**

Belgenin ne zaman ve neyden üretildiği, arkasındaki her sorgu ve döndürdüğü satır sayısı.

Üretildi 29 Oct 2026, 19:31 UTC — Üreten Attestkeep for Kubernetes 1.4.0

Kaynak sistem Attestkeep PostgreSQL store, installation e8845334a328351b0b4410fd8536e4bdbc8546a8797a6fa277ba7baa2257c41a — Sorgu penceresi [2026-10-01T00:00:00Z, 2026-10-30T00:00:00Z) UTC

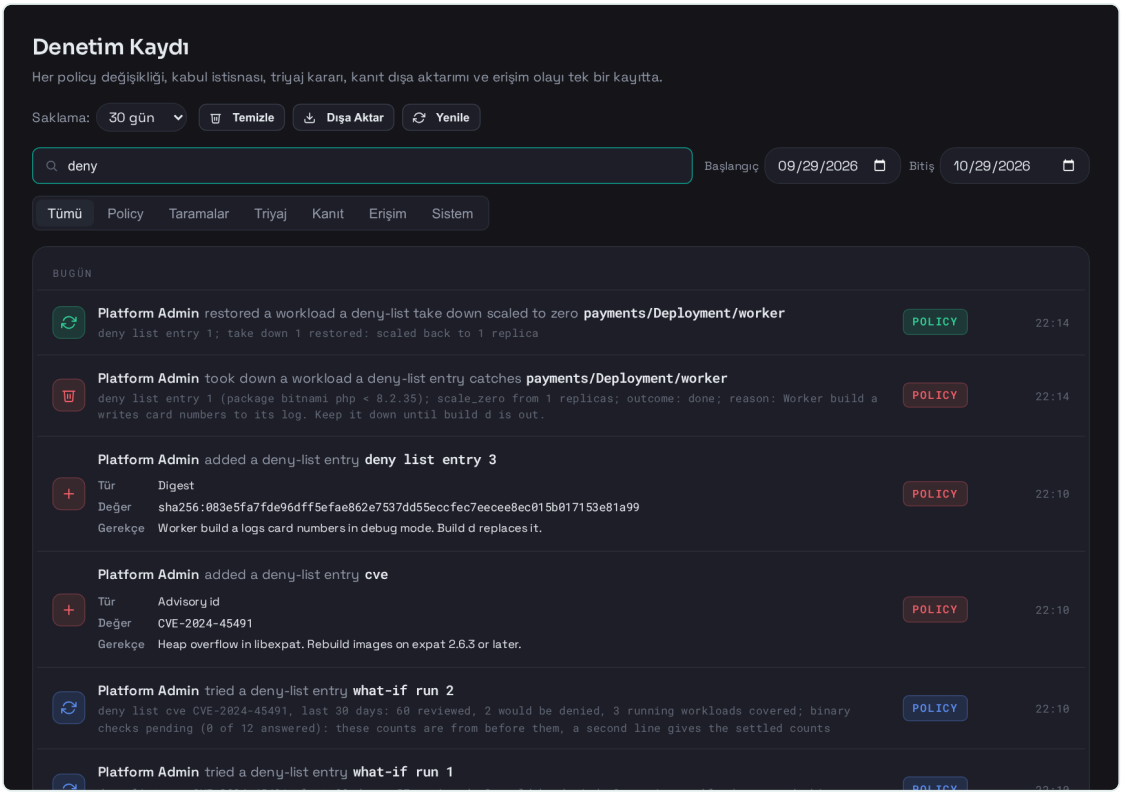
Kişiler user:<account id> · 0 satır bir hesaba eşlenemedi

BÖLÜM	SORGU	SATIR	KESİLDİ
admission.records	every row of admissions with at in [period.start, period.end)	125	—
admission.policy_revisions	the policy hashes of those rows, grouped, each joined to policy_revisions	3	—
chain (audit rows)	every row of audit_log with at in [period.start, period.end)	97	—
triage	every row of triage created before period.end, expired decisions included	0	—
availability.outages	every row of webhook_outages that started before period.end and had not ended by period.start	1	—
outcome.images	every row of images joined to its scan, merged to one row per name and tag, as the inventory stands at generation	12	—
findings.critical	every untriaged critical finding in the inventory at generation	268	—
vex.suppressed	every finding a trusted producer statement took out of the counts, at generation	0	—
issues	every row of issue_history opened in [period.start, period.end), runtime rows counted apart	0	—
population	the newest row of reconciliations before period.end	1	—
ledger	every row of ledger_checkpoints, and every admission row they seal, re-hashed	1	—
runtime.audit_entries	the audit rows above whose kind is runtime_rule_change, runtime_pod_stop or runtime_integration_change	2	—
runtime.decisions	rows of runtime_events with event_time in [period.start, period.end) and action notify (or block, as events were decided before 1.3.1), oldest first, at most 500	1	Hayır
external_export	every row of siem_targets with its siem_delivery_state, and every row of siem_give_ups with at in [period.start, period.end), grouped by destination	2	—
deny_list	every row of deny_list_entries, the rows of deny_list_hits first seen before period.end and last seen at or after period.start, every row of deny_list_takedowns made or restored in [period.start, period.end), and the admission.records above that name a deny_list_entry_id	9	—

Kişiler hesap id'siyle (user: ve ardından id) yazılır, asla ad ya da e-postayla; operatör id'yi konsol audit kaydından çözer. k8s-user: ve ardından bir hash, takma adlı bir Kubernetes kimliğidir, admission ledger'dan çözülür.

Bu kitapta gösterilen pakette *Bu belge nasıl üretildi* bölümü, 1.4.0 ile üretildi. Üretim zamanını, üreticiyi, kaynak sistemi, sorgu penceresini (bitiş hariç, UTC), kişilerin nasıl adlandırıldığını ve her bölümün sorgusunu döndürdüğü satırlarla verir. Son iki satır, `external_export` ve `deny_list`, 1.4.0'da geldi.

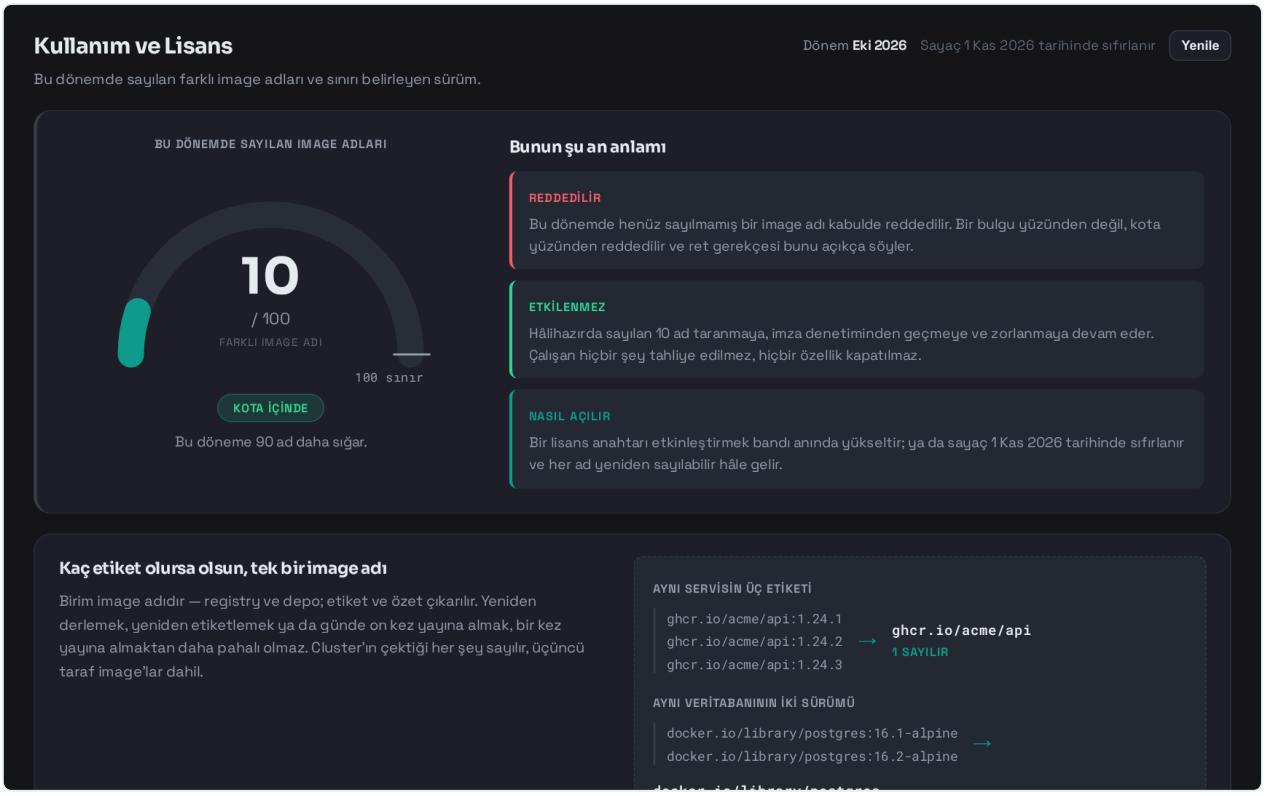
Denetim kaydı



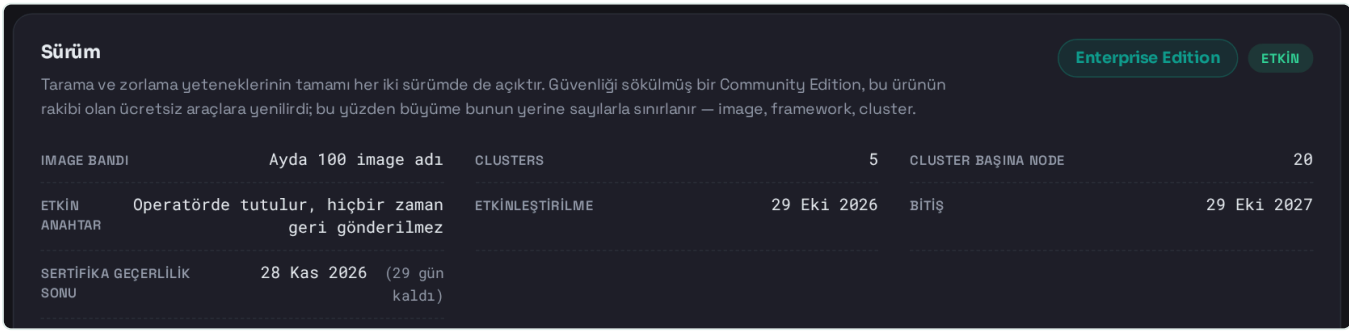
deny diye aranmış denetim kaydı. En yeni satırlar bir geri açma ve ondan önceki take down, türü, değeri ve gerekçesiyle eklenen iki kayıt, ve bir kaydın iki Ne olurdu koşusu. Aktöre, policy'ye, image'a ya da CVE'ye göre arar, Policy, Taramalar, Triyaj, Kanıt, Erişim ya da Sistem kategorisine göre süzersiniz.

Bildirim kanalı değişiklikleri ve testleri, Falco entegrasyonu değişiklikleri, runtime policy değişiklikleri, runtime policy'sinin durdurduğu pod'lar, rule editöründeki doğrulama, kaydetme ve yayınlama işlemleri, What if koşuları ve indirmeleri, Şimdi tara ve Şimdi güncelle burada kayda geçer. 1.4.0'dan itibaren deny list kaydı ekleme ve kaldırma, take down, geri açma ve yeniden kontrol (`deny_list_add` , `deny_list_remove` , `deny_list_takedown` , `deny_list_restore` , `deny_list_check_again`) ile SIEM hedefi ekleme, değiştirme, test etme ve kaldırma (`siem_target_add` , `siem_target_update` , `siem_target_test` , `siem_target_remove`) da yazılır. Saatler yerel saatinizle gösterilir, CSV dışa aktarımı ISO zaman damgalarını korur. Kanal işlemleri kanıt paketine girmez, çünkü bir image hakkında verilen kararı değil, operatörün kurulumda yaptığı işlemi anlatır.

Kullanım ve Lisans



Bu dönem sayılan farklı image adları: 100'de 10, kota içinde, yeni adlar reddedilmeden önce 90 ad daha var. Sayfa sınırda ne olacağını anlatır. 1.4.0'dan itibaren kota yüzünden reddedilen ad sayılmaz ve dolan bant *Kota doldu* der.



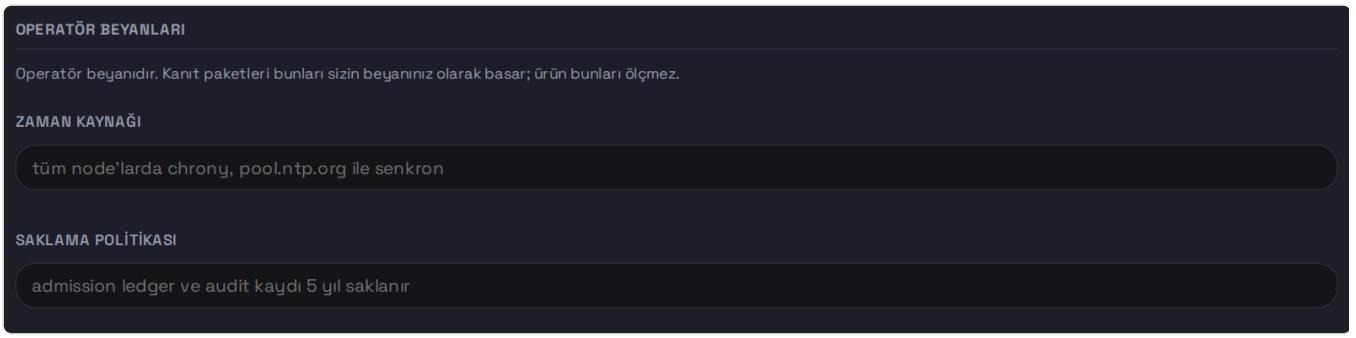
Bu kurulumun *Sürüm* bölümü: Enterprise Edition, etkin. Ayda 100 image adı, 5 cluster ve cluster başına 20 node. Lisans 29 Ekim 2026'dan 29 Ekim 2027'ye kadar sürer. Operatörün tuttuğu sertifika 28 Kasım 2026'ya kadar geçerli. Çevrimiçi bir sertifika en çok otuz gün geçerlidir ve günlük lisans kontrolü onu süresi dolmadan yeniler (12. bölüm).

Sayılan birim image adıdır: registry ve repository, tag ve digest atılmış haliyle. Günde on kez yeniden tag'lemek ya da göndermek, bir kez göndermekle aynı sayılır. 1.3.1'den itibaren bu sayfa ve panel içinde bulunulan ayı yürürlükteki lisansa göre gösterir. Lisans bitince, admission'ın uyguladığı Community hakları görünür.

1.4.0'dan itibaren aylık kota yüzünden reddedilen image adı kotaya sayılmaz. Sayaç limitte durur ve örneğin 8'de 8 ile *Kota doldu* yazar. 1.3.7 reddedilen adları limitin ötesinde saymaya devam ediyordu. Ret ve admission kaydı değişmedi.

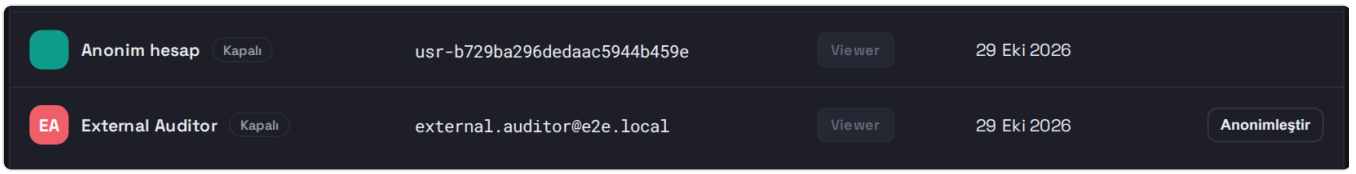
Şirket ve kullanıcılar

Ayarlar > Şirket, kanıt paketlerinin kurulumun arkasındaki kuruluş hakkında yazdıklarını tutar. *Operatör beyanları* altında zaman kaynağınızı ve saklama politikanızı yazarsınız. Buradaki saklama politikası kuruluşunuzun yazılı kuralıdır, ürünün policy nesnesi değildir. Konsol da alanı *Saklama politikası* diye anar. Ürün ikisini de ölçmez, paketler ikisini de operatörün beyanı olarak basar.



İki beyan alanı. Bu kurulumda boşlar. Gri metin, siz yazana kadar formun gösterdiği örnektir.

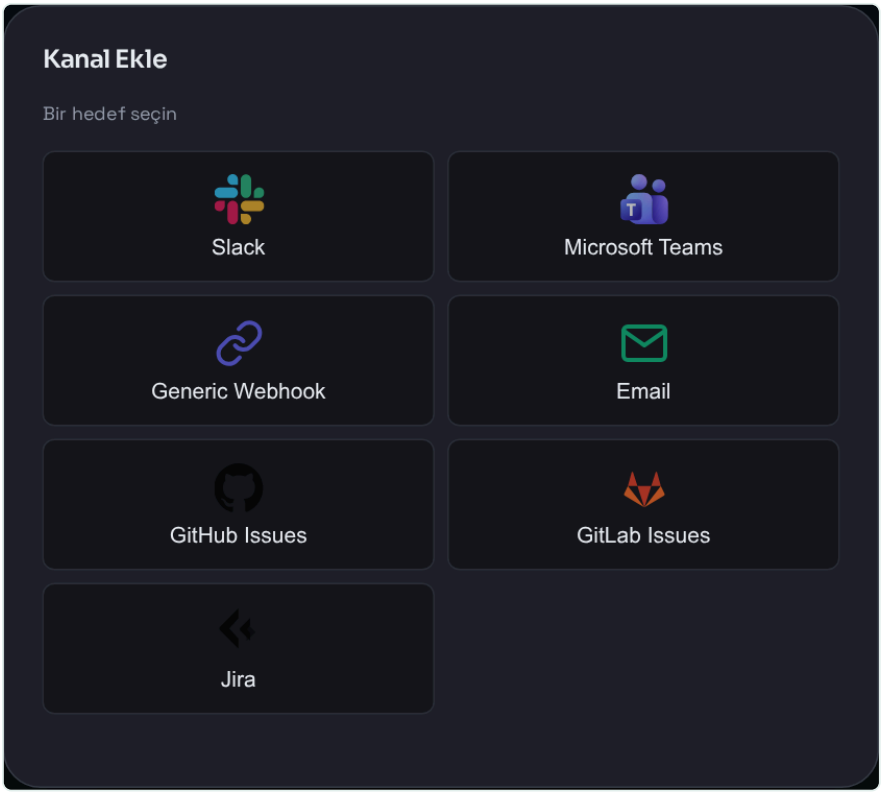
Kullanıcı silmek hesabı kaldırmaz, kapatır. Kapatılan hesap giriş yapamaz, oturumları sonlanır. Hesap id'si ve audit satırları kalır, böylece verilmiş paketlerdeki her `user:<id>` hâlâ bir hesaba karşılık gelir. Silme talebinde bir admin kapatılmış hesabı *Anonimleştir* ile anonimleştirebilir. Hesabın adı, e-postası ve avatarı silinir. Audit log'da, triage kararlarında, runtime policy'lerinde ve onların revizyonlarında geçen adı `user:<id>` olarak yeniden yazılır. Daha önce imzalanmış paketler imzalandıkları haliyle kalır, issue tracker'a yazılmış yorumlar değişmez.



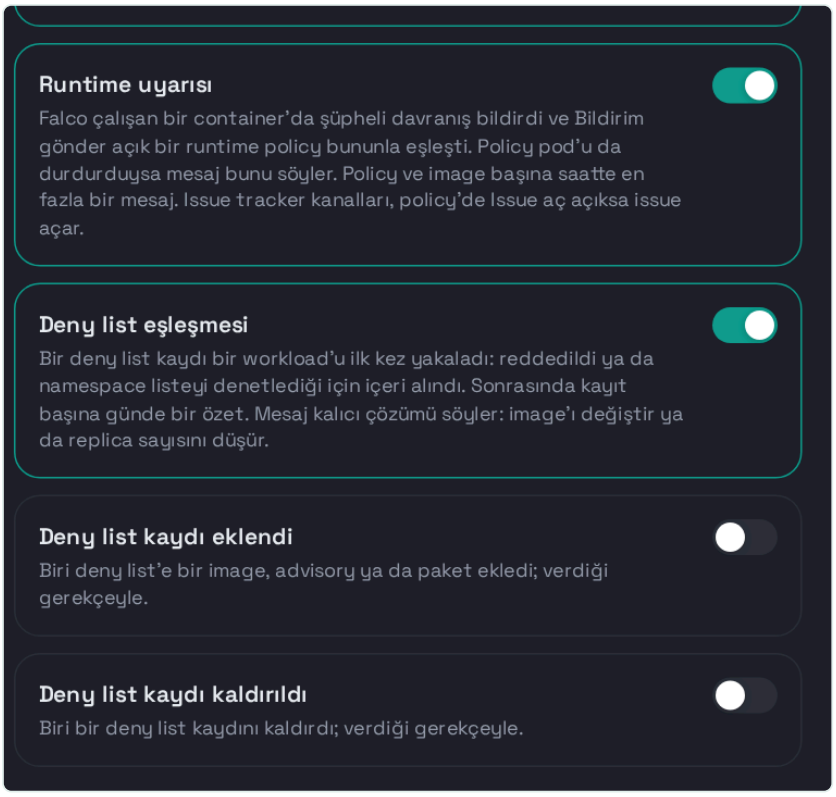
Anonimleştirilen hesap listede kapalı olarak kalır, e-posta adresinin yerinde hesap kimliği görünür. Altındaki kapalı hesapta *Anonimleştir* hâlâ duruyor.

Bildirim kanalları

Kanallar Ayarlar > Webhooklar'daki *Bildirim Kanalları* sayfasında kurulur. Herkesin kendi e-posta tercihleri Ayarlar > Bildirimlerim'dedir.



Kanal Ekle. Issue tracker da diğerleri gibi bir kanaldır, abonelikleri aynıdır. Tracker kanalında *Test* hiçbir şey yazmaz. Repo ya da projeyi bir kez okur ve kimlik bilgisinin çalışıp çalışmadığını söyler.



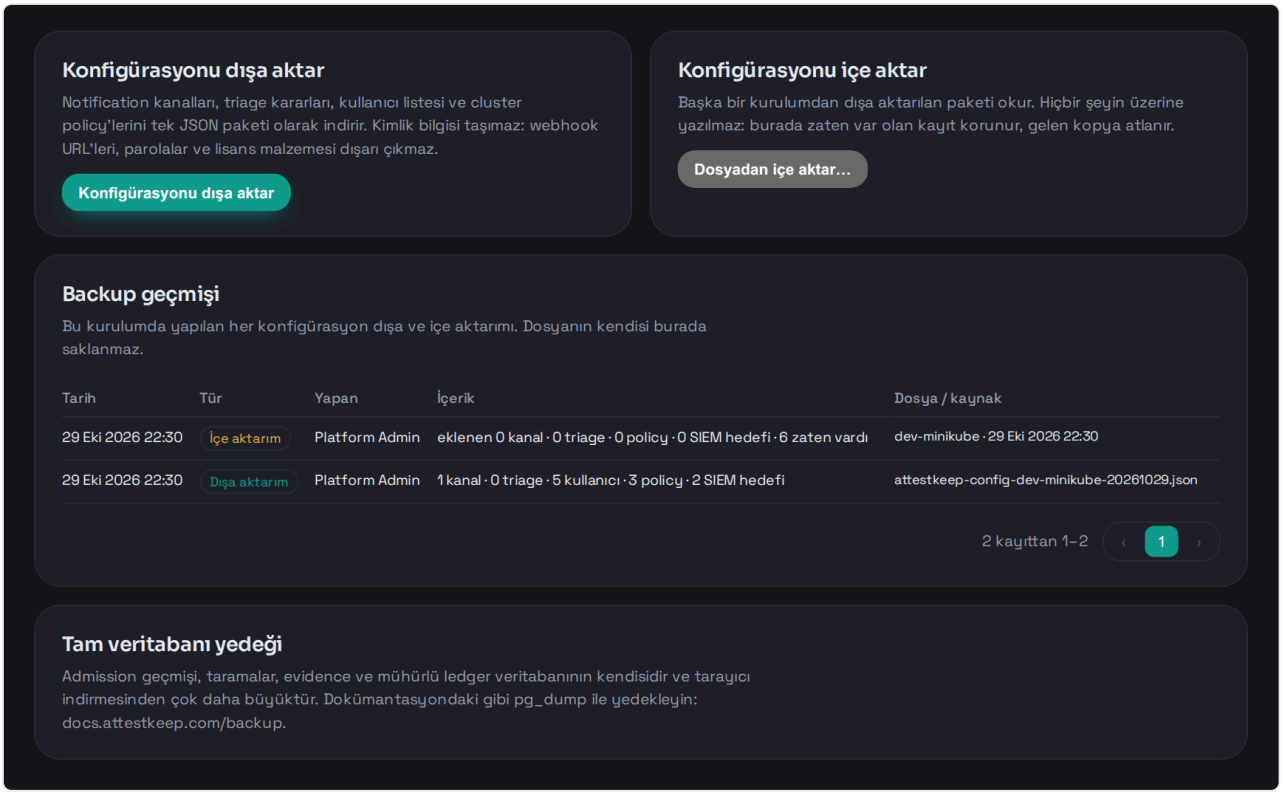
Bir kanalın *Event'ler* listesinden bir parça. 1.4.0'dan itibaren açılan her kanalda olduğu gibi *Deny list eşleşmesi* açık: bir kayıt bir workload'u ilk yakaladığında mesaj, sonra kayıt başına günde bir özet. *Deny list kaydı eklendi* ve *Deny list kaydı kaldırıldı* varsayılan olarak kapalı. *Runtime uyarısı* 8. bölümde anlatılıyor.

1.4.0'dan itibaren kanal deny list'ten de haber alabilir. *Deny list eşleşmesi*, bir kayıt bir workload'u ilk kez yakaladığında kaydı, gerekçesini ve çıkış yolunu söyler. Sonrasında kayıt başına günde bir özet gelir. Yeni kanalda işaretli gelir, 1.4.0'dan önce açılmış kanal aboneliklerini korur. Deny list reddi ayrıca *Admission blocked* olarak gönderilmez, o olay policy'nin kendi kurallarının retleri içindir. *Deny list kaydı eklendi* ve *Deny list kaydı kaldırıldı* varsayılan olarak kapalıdır.

E-posta, webhook, Slack ya da Teams ile her gönderimin bağlantıdan son bayta kadar 30 saniyesi vardır. Süresi dolan gönderim bildirim log'unda "no answer within 30s" ile ulaşmadı olarak kaydedilir. Aynı anda en fazla 64 gönderim çalışır ve en fazla 64 bildirim bekler. Fazlası bırakılır, ulaşmadı olarak kaydedilir ve `attestkeep_notifications_dropped_total` ile sayılır.

Issue tracker token'ı, Attestkeep'in cluster dışındaki bir sisteme yazabilen tek kimlik bilgisidir. API'den geri okunmaz. Saklanmadan önce AES-256-GCM ile şifrelenir. Anahtar chart'ın `<release>-data-key` Secret'ında durur, veritabanına yazılmaz. Anahtarı kendi secret yöneticinizden vermek için `api.dataKey.existingSecret` kullanın. Anahtar yoksa tracker kanalı kaydedilemez, başka hiçbir şey etkilenmez.

Backup geçmişi



Ayarlar > Yedekleme. *Konfigürasyonu dışa aktar* bildirim kanallarını, triyaj kararlarını, kullanıcı listesini, policy'leri ve 1.4.0'dan itibaren SIEM hedeflerini kimlik bilgisi olmadan bir JSON paketi olarak indirir. *Backup geçmişi* her dışa ve içe aktarmayı ne zaman, kimin yaptığı ve paketin ne taşıdığıyla listeler. Operatör paket dosyasını saklamaz. Veritabanı yedeği sizin işinizdir (11. bölüm).

6. Admission policy'leri

Neyin çalışabileceğine cluster kapsamlı tek bir kaynak karar verir: `attestkeep.com` grubundaki `ImageSecurityPolicy`. Konsol bu nesneyi doğrudan düzenler, `kubectl apply` repository'den düzenler. İkisi aynı nesne üzerinde çalışır. Değişiklik bir sonraki admission kararında geçerli olur, restart gerekmez.

Namespace'in policy'si. `spec.targetNamespaces` listesinde o namespace'i sayan policy uygulanır. Yoksa `targetNamespaces` alanı olmayan, yani genel policy uygulanır. Yaygın düzen bir genel policy ve sıkı kontrol gereken namespace'ler için ikinci bir policy'dir.

Chart'la gelen varsayılan

```
apiVersion: attestkeep.com/v1alpha1
kind: ImageSecurityPolicy
metadata:
  name: default
spec:
  enforcement: Enforce          # Audit yalnız neyi reddedeceğini kaydeder
  admissionTiming: AllowAndScan # DenyUntilScanned tarama kaydı olmayanı reddeder
  blockLatestTag: true          # değişebilen tag, deploy olmadan çalışanı değiştirir
  digestEnforcement: AllowTags  # AllowTags | PreferDigest | Required
  denyList: "enforce"           # enforce | audit | "off" (7. bölüm)
  thresholds:
    blockCritical: true
    blockHigh: true
    maxMedium: 0                # burada 0 sınırsız demek
    maxLow: 0
  slaWindows:
    critical: 24h
    high: 168h
    medium: 720h
    low: 2160h
  requireSignature: false       # pipeline imza üretmeye başlayınca açın
  requireSBOM: false
  requireVulnAttestation: false
  attestationMaxAge: 168h
  triageDefaults:
    maxDuration: 2160h
    requireJustification: true
```

Production namespace'leri için daha sıkı bir policy

```
apiVersion: attestkeep.com/v1alpha1
kind: ImageSecurityPolicy
metadata:
  name: prod-hard-gate
spec:
  targetNamespaces: ["prod", "payments"]
  enforcement: Enforce
  admissionTiming: DenyUntilScanned
  blockLatestTag: true
  digestEnforcement: Required
  allowedRegistries:
    - registry.example.com
    - ghcr.io/your-org
  thresholds:
    blockCritical: true
    blockHigh: true
  requireSignature: true
  requireSBOM: true
  requireVulnAttestation: true
  attestationMaxAge: 168h
  workloadHardening:
    mode: enforce              # off | warn | enforce
    minSeverity: high
    exempt: []
```

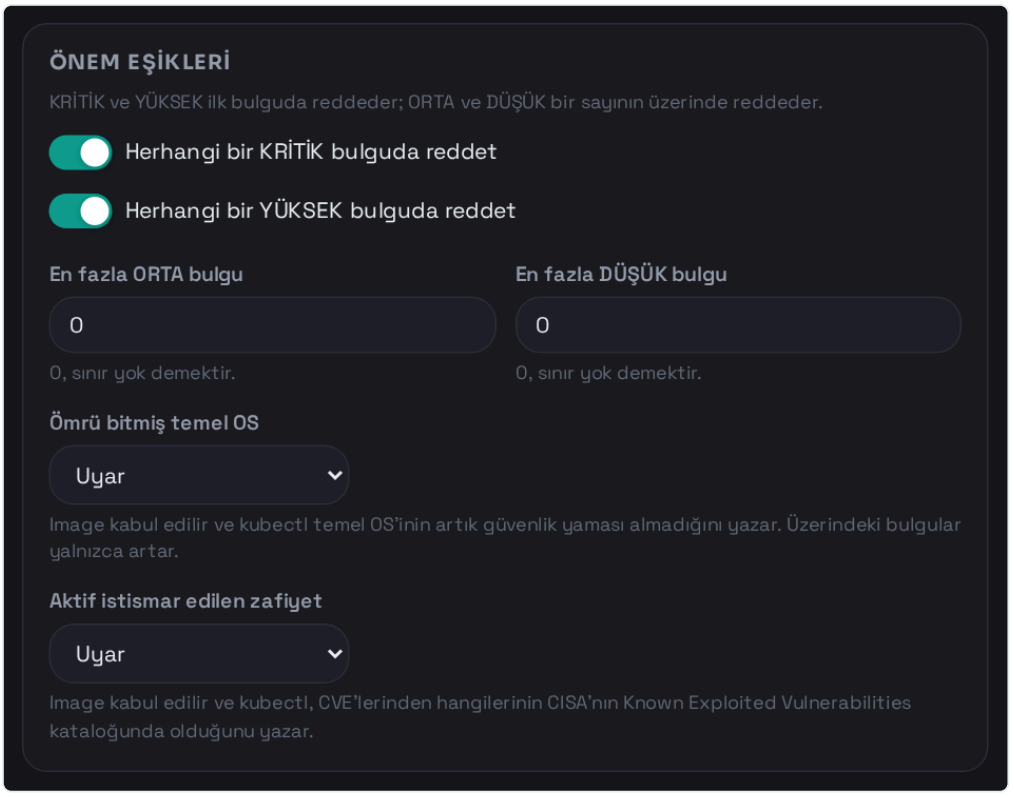
Desteği bitmiş base OS ve istismar edilen CVE'ler

İki kural severity sayısı dışında bir şeye bakar. İkisi de `ignore`, `warn` ya da `block` alır, varsayılanları `warn`'dır. Bu alanlar eklenmeden önce yazılmış bir policy eskiden kabul ettiğini kabul etmeye devam eder, üstüne neden etmemesi gerektiğini de yazar.

```
spec:
  # Base OS, üreticisinin destek süresini geçmiş; bu sürüm için artık
  # güvenlik yaması çıkmayacak. warn (varsayılan) kabul eder ve yazar;
  # block sayılara bakmadan reddeder; ignore hiçbir şey yazmaz.
  endOfLifeOS: warn

  # Image, CISA'nın Known Exploited Vulnerabilities kataloğundaki bir CVE'yi
  # taşıyor. warn (varsayılan) kabul eder ve CVE'leri yazar; block reddeder;
  # ignore hiçbir şey yazmaz.
  knownExploited: warn
```

KEV kataloğu yerelde tutulur ve altı saatte bir yenilenir. İnternete çıkışı olmayan cluster `kev.source` ile bir mirror gösterir ya da elindeki kopyayla devam etmek için alanı boş bırakır. Triage edilmiş CVE'ler bu kuraldan da severity eşiklerinden olduğu gibi muaftır. Ret mesajı CVE'leri tek tek yazar, böylece deploy eden ne düzelteceğini bilir. Panel kullanılan katalog sürümünü gösterir, kanıt paketi bu sürümü zafiyet veritabanı sürümünün yanına yazar.



Editörün önem eşikleri bölümü. Ömrü bitmiş temel OS ve aktif istismar edilen zafiyet ayarlarının ikisi de *Uyar*'a ayarlı. Her birinin altındaki metin o ayarın ne yaptığını söyler.

İmza anahtarları

1.4.0'dan itibaren policy kimin imzasını kabul ettiğini anahtarla söyleyebilir. `signatureKeys`, örneğin `cosign.pub` dosyanız gibi PEM public key'leri listeler. `requireSignature: true` ile yalnız bu anahtarlardan birinin doğruladığı imza sayılır.

```
spec:
  requireSignature: true
  signatureKeys:
    - |
      -----BEGIN PUBLIC KEY-----
      MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE...
      -----END PUBLIC KEY-----
```

Tarama imzalayanı anahtarın parmak iziyle, `key sha256:…` biçiminde kaydeder, kanıt paketi de bu adı yazar. `scan.cosignOffline` ile kontrol transparency log'a ihtiyaç duymaz. SBOM ve zafiyet attestation'ları keyless kalır, anahtar alanları yoktur. API server PEM public key biçiminde olmayan girdiyi reddeder. Biçimi doğru olup yine de okunamayan girdi yok sayılır ve üç yerde adıyla yazılır: policy'nin `SignatureKeysValid` status koşulu, bir Warning event'i ve policy sayfası.

Build provenance

Bu kural image'ı hangi build sisteminin hangi kaynaktan ürettiğine bakar. Bir kimliğe bağlanan tek kontrol budur. `requireSBOM` ve `requireVulnAttestation` attestation'ı birinin imzaladığını kontrol eder, bu kural kimin imzaladığını kontrol eder.

```
spec:
  provenance:
    # 0 (varsayılan; blok hiç yoksa da 0) kuralı kapatır. 0'ın üstünde
    # image, aşağıdaki listeden bir builder'ı adlandıran doğrulanmış SLSA
    # provenance taşımalı ve builder'ın level'ı en az bu olmalı.
    minLevel: 2
    trustedBuilders:
      # Sondaki "*" önekle eşleşir, diğer id'ler birebir eşleşir.
      # minLevel 0'ın üstündeyken boş liste hiçbir şeyi kabul etmez.
      - id: https://github.com/acme/build/.github/workflows/release.yml@*
        # Bu builder hakkındaki sizin beyanınız. SLSA provenance level
        # taşımaz, level builder'ın nasıl işletildiğini anlatır.
        level: 3
        # İmza sertifikasından okunur. Boşsa builder id ile aynı olmalı.
        # issuer boşsa her issuer kabul edilir.
        signer: https://github.com/acme/build/.github/workflows/release.yml@*
        issuer: https://token.actions.githubusercontent.com
```

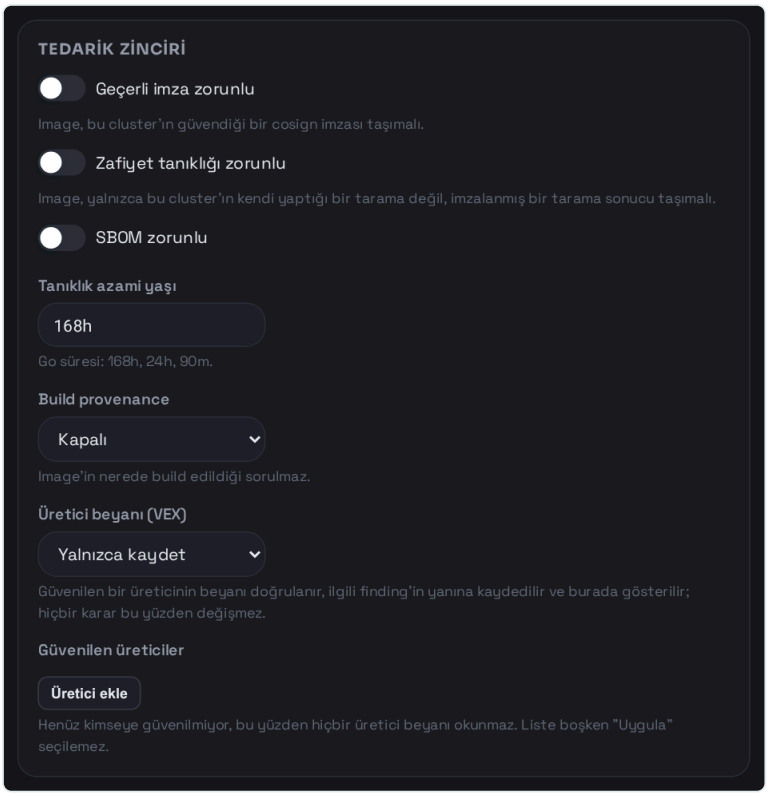
Doğrulanmış SLSA provenance'ı (v1 ya da v0.2) güvenilen bir builder'ı adlandırmayan, imza sertifikasındaki kimliği ya da issuer'ı girdiyle uyuşmayan veya builder level'ı `minLevel`'ın altında kalan image reddedilir. Gerekçe builder'ı, signer'ı ya da level'ı yazar. İki kimlik de attestation'ın yanında saklanan sertifikadan okunur. Sertifika alınamazsa ya da okunamazsa image reddedilir.

Üretici beyanları (VEX)

Scanner image'da neyin kurulu olduğunu söyler. Zafiyetli kodun erişilebilir olup olmadığını ya da hiç çalışıp çalışmadığını söyleyemez. Bunu image'ı üreten taraf söyleyebilir, bunun biçimi de OpenVEX'tir. Attestkeep VEX beyanını yalnız image üzerinde imzalı bir in-toto attestation olarak geldiğinde ve policy imzalayanı tanıdığında okur. İmzasız beyanlar, policy'nin tanımadığı imzalayanlardan gelenler ve başka ürüne ait beyanlar kaydedilir ama dikkate alınmaz.

```
spec:
  vex:
    # ignore (varsayılan): güvenilen beyanlar doğrulanır, ilgili bulguya
    # yazılır ve konsolda görünür; hiçbir karar değişmez.
    # apply: güvenilen bir not_affected ya da fixed beyanı o bulguyu
    # admission'ın baktığı sayılardan çıkarır.
    mode: apply
    trustedIssuers:
      # Sertifika kimliğiyle keyless imzalayan. Sondaki "*" önekle eşleşir.
      # issuer isteğe bağlı, birebir eşleşir.
      - identity: https://github.com/acme/images/*
        issuer: https://token.actions.githubusercontent.com
      # Ya da public key ile anahtarlı imzalayan. Bir girdi identity ya da
      # key taşır, ikisini birden taşımaz.
      - key: |
          -----BEGIN PUBLIC KEY-----
          MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE...
          -----END PUBLIC KEY-----
```

`affected` ve `under_investigation` hiçbir bulguyu sayımdan çıkarmaz. Çıkarılan bulgu severity eşiklerinin ve KEV kuralının baktığı sayılardan düşer, izin gerekçesine kaç bulgunun kimin beyanıyla çıkarıldığı yazılır. Boş `trustedIssuers` ile `mode: apply` reddedilir, çünkü kontrol gibi görünüp hiçbir şey yapmaz. Mod değişikliği bir sonraki pod'da, yeni issuer ise digest'in bir sonraki taramasında geçerli olur.



Tedarik zinciri bölümü. İmza, zafiyet attestation'ı ve SBOM şartları kapalı, build provenance *Kapalı*, *Üretici beyanı (VEX)* varsayılan değer olan *Yalnızca kaydet*'te. Henüz güvenilen bir yayımcı yok, bu yüzden hiçbir beyan okunmuyor.

Pod'un nerede çalışacağı

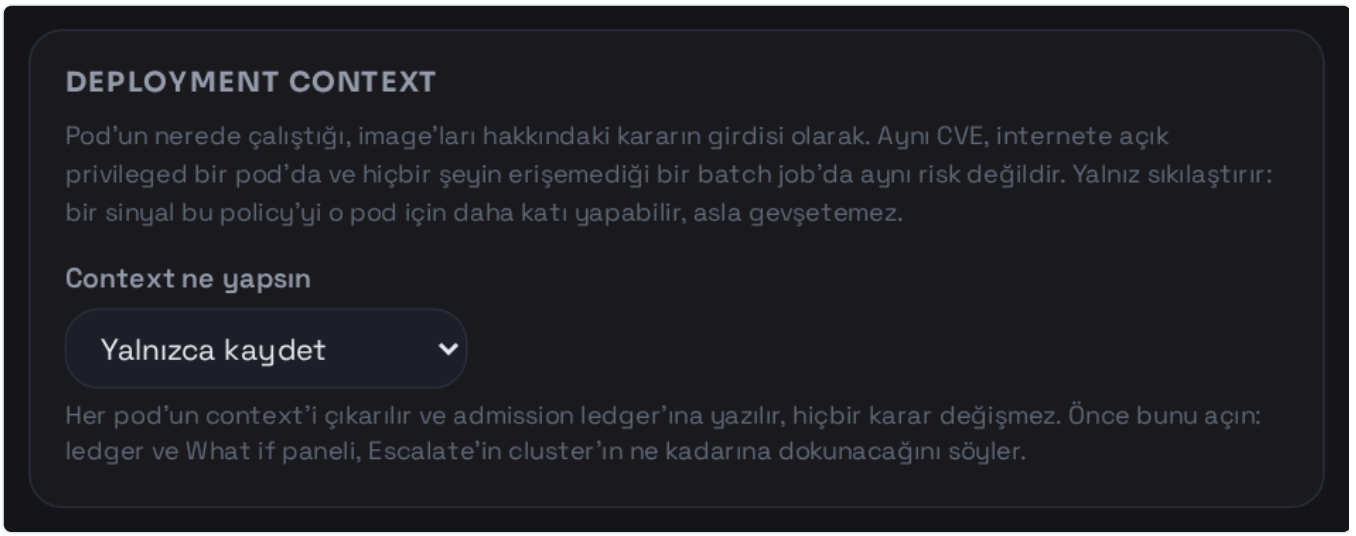
Aynı HIGH bulgu, internete açık privileged bir pod'da hiçbir yerden erişilemeyen bir batch job'dakinden daha fazla risk taşır. `deploymentContext` bu tür pod'ları daha sıkı değerlendirmeyi sağlar. Kararı yalnız sıkılaştırır. Hiçbir sinyal ya da sinyal kombinasyonu, policy'nin reddedeceği bir pod'u kabul ettirmez.

```
spec:
  deploymentContext:
    mode: Escalate      # Ignore (varsayılan) | Escalate
    signals:            # boşsa altısı da
      - exposed
      - privileged
```

Her pod için altı sinyal çıkarılır ve iki modda da her ledger satırına yazılır: `privileged`, `root`, `hostNamespace`, `hostFilesystem`, `capabilities` ve `exposed`. Pod'un dışına bakan tek sinyal `exposed`'dır: hostNetwork, hostPort, pod'u seçen NodePort ya da LoadBalancer Service veya Ingress arkasındaki ClusterIP Service. Önünde Ingress olmayan ClusterIP Service dışarıya açık sayılmaz. `Escalate` modunda seçili sinyali taşıyan pod şöyle değerlendirilir:

Policy'de	Seçili sinyali taşıyan pod için
`thresholds.blockCritical: true`	HIGH da reddedilir
`thresholds.blockHigh: true`	MEDIUM da reddedilir
`knownExploited: warn`	`block`
`endOfLifeOS: warn`	`block`

`maxMedium` ve `maxLow` değişmez, `ignore` da `ignore` kalır. Ret mesajı sinyali, exposure için de buna yol açan Service'i, Ingress'i ya da hostPort'u yazar.



Deployment context bölümü *Yalnızca kaydet*'e, yani `Ignore`'a ayarlı. Her pod'un context'i admission ledger'ına yazılır ve hiçbir karar değişmez. Ledger ve Ne olurdu paneli sonra `Escalate`'in kaç pod'a dokunacağını gösterir.

Alan listesi

Alan	Değerler	Neyi belirler
enforcement	Enforce | Audit	Reddetmek ya da yalnız reddedeceğini kaydetmek.
admissionTiming	AllowAndScan | DenyUntilScanned	Tarama kaydı olmayan image'a ne olacağı.
blockLatestTag	bool	`:latest` tag'ini reddetmek.
digestEnforcement	AllowTags | PreferDigest | Required	Image'ların digest ile sabitlenmesinin şart olup olmadığı.
allowedRegistries	host listesi	Boşsa her registry kabul. Doluysa diğerleri reddedilir.
denyList	enforce | audit | "off"	Deny list eşleşmesinin bu policy'nin namespace'lerinde ne yapacağı. Varsayılan `enforce`, `enforcement: Audit` altında bile reddeder (7. bölüm).
targetNamespaces	liste	Boşsa genel policy. Doluysa yalnız o namespace'ler.
thresholds	blockCritical, blockHigh, maxMedium, maxLow	Kritik ya da yüksek bulguda ret. Orta ve düşük için tavan (0 sınırsız).
slaWindows	süreler	Her severity'deki bulgunun raporlarda gecikmiş sayılmadan ne kadar açık kalabileceği.
requireSignature / requireSBOM / requireVulnAttestation	bool	Doğrulanmış cosign imzası, SBOM attestation'ı ya da zafiyet taraması attestation'ı olmayan image'ı reddetmek.
signatureKeys	PEM public key listesi	`requireSignature` ile yalnız bu anahtarlardan birinin doğruladığı imza kabul edilir.
attestationMaxAge	süre	Bundan eski attestation sayılmaz.
workloadHardening	mode, minSeverity, exempt	Privileged container, hostPath mount ve benzerlerinde ret, uyarı ya da geçiş.
endOfLifeOS	ignore | warn | block	Base OS'i destek süresini geçmiş image'lar. Varsayılan `warn`.
knownExploited	ignore | warn | block	CISA KEV kataloğundaki bir CVE'yi taşıyan image'lar. Varsayılan `warn`.
provenance	minLevel 0–3, trustedBuilders (id, level, signer, issuer)	Güvenilen builder'dan, istenen level'da doğrulanmış provenance'ı olmayan image'ı reddetmek. Varsayılan `minLevel: 0`, kapalı.
vex	mode ignore | apply, trustedIssuers	Güvenilen üreticinin not_affected ya da fixed beyanının bulguyu sayımdan çıkarıp çıkarmayacağı. Varsayılan `ignore`.
deploymentContext	mode Ignore | Escalate, signals	Seçili sinyali taşıyan pod'u daha sıkı değerlendirmek. Varsayılan `Ignore`.
runtimeReconciliation	audit | notify | enforce	Arkasında admission olmayan çalışan image için periyodik taramanın ne yapacağı.
breakGlass	enabled, reason, approvedBy, auditRef, expiresAt, affectedImages	Adı ve onayı olan, süresi dolan istisna. Her kullanımı kaydedilir.

triageDefaults	maxDuration, requireJustification	Kabul edilen riskin ne kadar geçerli kalacağı ve gerekçe isteyip istemediği.
storageUnreachablePolicy	Allow | Deny	Veritabanına ulaşılamazken admission cevabı.

Bir Falco event'inin neye yol açacağı admission policy'sinde yazmaz. Bunu konsoldaki runtime policy'leri belirler (8. bölüm).

Break-glass

Production durmuşsa ve çözüm policy'nin reddettiği bir image ise, break-glass adı verilen image'ları içeri alır, her kullanımı kaydeder ve süresi dolunca kendiliğinden kapanır. Konsoldaki karşılığı *Acil erişim* düğmesidir.

```
breakGlass:
  enabled: true
  reason: "INC-2417: geri dönüş image'ı bilinen bir high bulgu taşıyor"
  approvedBy: "nöbetçi, olay sorumlusu"
  auditRef: "INC-2417"
  expiresAt: "2026-09-01T06:00:00Z"
  affectedImages:
    - registry.example.com/payments/api:1.41.2
```

Yalnız break-glass sayesinde kabul edilen her admission, audit kaydında ve compliance raporlarında böyle işaretlenir.

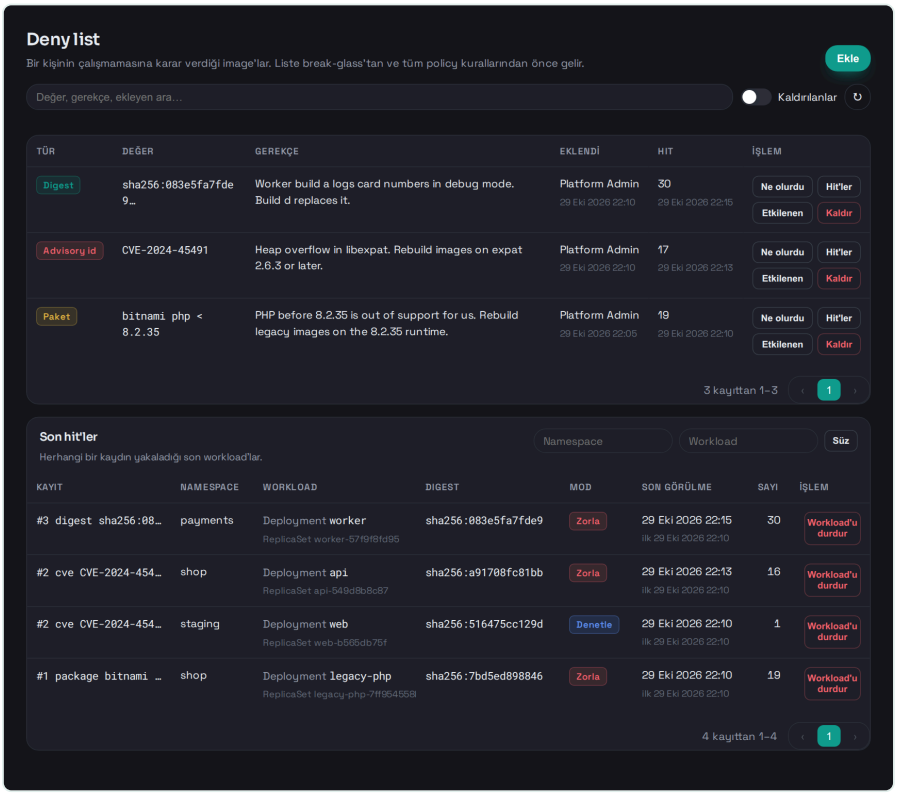
Break-glass'ın kapsadığı image'lar policy'nin image kuralları uygulanmadan kabul edilir: registry izin listesi, `:latest` ve digest kuralları, tarama sonucu ve onun kuralları, imzalar, attestation'lar ve build provenance, image kotası ve node tavanı. Deny list ve workload hardening uygulanmaya devam eder. `affectedImages` image adıyla eşleşir: `nginx`, `docker.io/library/nginx` ve `nginx:1.27` aynı image'dır, listedeki tag kapsamı daraltmaz. Boş liste her image'ı kapsar. 1.4.0'dan önce break-glass açılırken liste saklanmıyordu, bu yüzden belirli image'larla sınırlanan pencere policy'nin bütün image'larını kapsıyordu.

7. Deny list

Policy bir soruyu kendiliğinden cevaplar: bu image kurallara uyuyor mu. Deny list kaydı başka bir soruyu cevaplar: biri bu image'ın çalışmamasına karar verdi mi. Deny list bu kararın yazıldığı yerdir ve admission ona her şeyden önce bakar.

Her cluster'da tek bir liste vardır, Güvenlik > Deny list altında. Her policy'nin `denyList` alanı, eşleşme olunca o policy'nin namespace'lerinde ne olacağını belirler. Kayıt hiçbir zaman silinmez. Kaldırmak kaydı devreden çıkarır, satır kimin, ne zaman ve neden kaldırdığıyla kalır. Ekleme ve kaldırma yönetici işidir, ikisi de gerekçe ister.

> **Boş liste hiçbir şeyi değiştirmez.** Etkin kayıt yokken admission fazladan digest çözmez ve hiçbir şeyi reddetmez. 1.4.0'a yükseltmek, biri kayıt ekleyene kadar hiçbir kararı değiştirmez.



Güvenlik > Deny list'te üç kayıt: bir digest, CVE-2024-45491 advisory'si ve `bitnami php < 8.2.35` paketi. Her satır gerekçeyi, kimin ne zaman eklediğini ve kaç hit aldığını verir. Alttaki *Son hit'ler* kayıtların yakaladığı workload'ları namespace ve digest'e göre adlandırır, namespace'in policy'sinin listeyi zorlayıp zorlamadığını ya da `staging`'deki gibi yalnız denetleyip denetlemediğini gösterir.

Kayıt türleri

Tür	Neyle eşleşir	Örnek
Digest	Tek bir image build'i. Admission'ın pod için çözdüğü digest'le ve bilinen image index ya da platform manifest'leriyle karşılaştırılır.	`sha256:9f2a…d4e5`
Advisory id	Son taraması bu advisory'yi bildiren her image, düzeltmesi olsun olmasın.	`CVE-2021-44228`, `GHSA-jfh8-c2jp-5v3q`
Paket	Son taraması paketi bu sürüm aralığında listeleyen her image.	`maven org.apache.logging.log4j:log4j-core < 2.17.1`
Dosya hash'i	Bir dosyayı, adı ve yolu ne olursa olsun, baytlarının sha256'sıyla taşıyan her image.	bir sha256
Dosya yolu	Bir yolda çalıştırılabilir dosya ya da herhangi bir yerde bu adda dosya taşıyan her image. Yol kuralı base image'ları da yakalar.	`/bin/sh`, `/usr/local/bin/*`, `nc`
Paket, `bitnami` ekosistemi	Çalıştırılabilir dosyanın içinden okunan sürümü aralıkta kalan runtime'ı taşıyan her image. Dağıtım paketiyle asla eşleşmez.	`bitnami php < 8.2.35`

Advisory kaydı, tarayıcıların bulgu bildirdiği şemaları kabul eder, çünkü bazı bulguların tek adı bu id'dir: CVE, GHSA, GO, PYSEC, RUSTSEC, OSV, MAL, GSD, DLA, DSA, ELA, USN, RHSA, RHBA, ALSA, ALBA, ELSA, CGA, BIT, TEMP, SUSE-SU ve openSUSE-SU. Başka bir şey kayıt eklenirken reddedilir, çünkü hiçbir tarayıcının bildirmediği bir id hiçbir şeyle eşleşmez.

Paket kaydı bir ekosistem, bir paket ve tek bir karşılaştırma alır: altında, en çok, tam olarak, en az, üstünde, arasında (alt sınır dahil, üst sınır hariç) ya da her sürüm. İki aralık iki kayıttır. Sürümler, tarayıcının o ekosistem için kullandığı karşılaştırıcıyla kıyaslanır: npm, Go, Cargo, Composer, NuGet, Maven, PyPI, RubyGems, deb, apk ve rpm. Karşılaştırıcının okuyamadığı sürüm, metin olarak kıyaslanmak yerine kayıt eklenirken reddedilir. Metin kıyasında 10.0, 9.0'ın altına düşerdi. *Üstünde* ya da *en az* ile yazılan kayıt, eklendikten sonra çıkan her sürümü de kapsar. What if bunu söyler.

Karardaki yeri

Sıra şöyledir: deny list, sonra break-glass, sonra policy'nin kuralları. Listedeki image taraması temiz olsa da, break-glass açıkken de reddedilir. Kayıt bilerek verilmiş bir karardır ve acil durum kapısından önce gelir. Ret şöyle okunur:

```
denied by deny list entry 7: package maven org.apache.logging.log4j:log4j-core < 2.17.1
```

Registry'ye gitmeyi gerektirmeyen kurallar, yani registry izin listesi, `blockLatestTag` ve digest zorunluluğu, önce bakılır. Enforce modundaki policy'de bunların reddi tek başına geçerlidir. Audit modundaki policy'de inceleme deny list'e devam eder, böylece listedeki image orada da reddedilir.

Policy'de `denyList`	Eşleşme o namespace'lerde ne yapar
`enforce` (varsayılan)	Pod'u reddeder, policy'nin enforcement'ı Audit olsa bile. Boş ya da hiç yazılmamış alan `enforce` sayılır.
`audit`	Pod'u *would be denied by deny list entry …* uyarısıyla kabul eder ve eşleşmeyi kaydeder.
`"off"`	Listeye bakılmaz.

YAML'da `off`'u tırnakla yazın, tırnaksız `off` YAML 1.1 okuyucusu için boolean'dır. Yükseltmeden sonra her policy bir yeni revizyon kaydeder, çünkü bu alan ledger'ın hash'lediği içeriğin parçasıdır.

Hangi isteklere bakar. Yeni bir pod'un her container'ına, init container'ına ve ephemeral container'ına. `kubectl debug`'da yalnız isteğin eklediği container'a. Mevcut pod'un label, annotation ya da finalizer gibi diğer güncellemeleri ne reddedilir ne sayılır. Dry run gerçek istek gibi cevaplanır ve eşleşme kaydetmez.

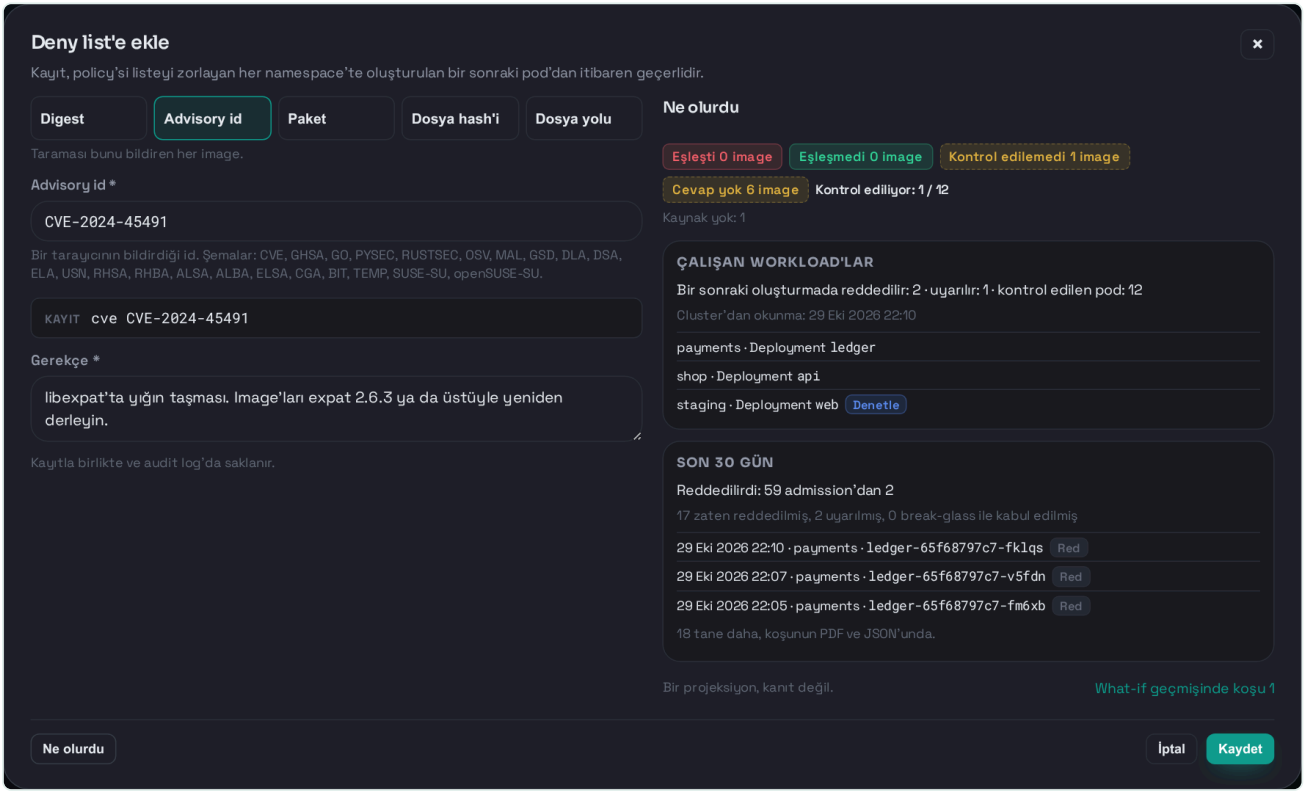
Çalışan hiçbir şey durmaz. Kayıt bir workload'u, o workload'un pod'u bir sonraki oluşturulduğunda reddeder. Workload'un controller'ı, image değişene kadar kendi bekleme süreleriyle denemeye devam eder.

Digest çözülemezse. Listede kayıt varken admission pod'un digest'ini break-glass'tan önce çözer. Registry cevap vermezse, `denyList: enforce` ayarlı enforce modundaki policy altındaki pod *deny list could not be evaluated: digest unresolved* ile reddedilir. Audit modundaki policy'de ya da `denyList: audit` ile aynı uyarıyla kabul edilir. Yalnız node'ların çekebildiği, yani pod'un ya da service account'unun referans verdiği bir pull secret yerine node kimlik bilgileriyle çekilen image'ı operator çözemez. Liste boş değilken böyle bir image enforce eden namespace'lerde reddedilir. Digest ile sabitleyin ya da pod'dan bir pull secret'a referans verin. Veritabanı okunamazsa liste uygulanmaz ve pod *deny list unavailable* uyarısını taşır.

Kayıt eklenmeden önce What if

What if (konsolda *Ne olurdu*), bir kaydı kimse admission'da öğrenmek zorunda kalmadan önce dener: Ekle penceresinde kaydetmeden önce ya da kaydın satırından. Yan yana iki şeyi cevaplar.

- **Kapsadığı çalışan workload'lar.** O anda cluster'dan okunur ve her pod'un ait olduğu workload'a göre gruplanır: ReplicaSet'in üstündeki Deployment, Job'ın üstündeki CronJob, bir StatefulSet, bir DaemonSet ya da pod'un kendisi. Büyük cluster'da okuma 20 saniye, 20.000 pod ya da 2.000 farklı image'dan sonra durur, sonucun kısmi olduğunu ve kaç pod incelendiğini yazar.
- **Reddedeceği geçmiş admission'lar.** Varsayılan olarak son 30 gün, en fazla 90: ledger satırları her namespace'in bugünkü policy'si altında, liste yalnız bu kayıttan oluşuyormuş gibi yeniden değerlendirilir. Break-glass ile kabul edilmiş admission'lar da değerlendirilir, çünkü break-glass deny list'i hiçbir zaman geçmez.



Ekleme penceresinde, kayıt kaydedilmeden önce *Ne olurdu*. Çalışan workload'lardan 2'si yeniden oluşturulduğunda reddedilir, 1'i `staging` listeyi denetlediği için uyarılır. Son 30 günde 61 admission'dan 2'si reddedilirdi, 17'si zaten reddedilmiş, 2'si uyarılmıştı. Dosya kontrolü hâlâ sürüyordu, 12 image'dan 3'ü. Çipler kaç image'ın henüz cevabı olmadığını gösterir. Koşu What-if geçmişinde 2 numaralı koşu olarak saklanır.

Hiçbir şey durdurulmaz, hiçbir şey kaydedilmez. Deployer ya da yöneticinin çalıştırdığı koşu Güvenlik > What-if geçmişi'nde *Deny list* türüyle tutulur: her koşu gibi doksan gün, deny list koşularından en yeni beş yüzü, policy koşularından ayrı

sayılarak. PDF ya da JSON olarak indirilir, kanıt değil projeksiyondur ve hiçbir zaman kanıt paketine girmez.

Eşleşmeler

Bir kaydın yol açtığı her ret ve her audit uyarısı workload'a sayılır: kayıt, namespace, workload ve digest başına bir satır, ilk ve son görülme zamanı ve kaç kez olduğuyla. Deployment'ın pod'ları Deployment'a, CronJob'ınkiler CronJob'a sayılır, böylece rollout yeni bir satır açmaz. Deny list sayfası listenin altında bütün kayıtların son eşleşmelerini gösterir. Satırdaki *Hit'ler* o kaydınkileri namespace ya da workload'a göre süzer. Her reddin admission ledger satırı kararı veren kaydı yazar.

Çalıştırılabilir dosyalar: tarayıcının listelemediği

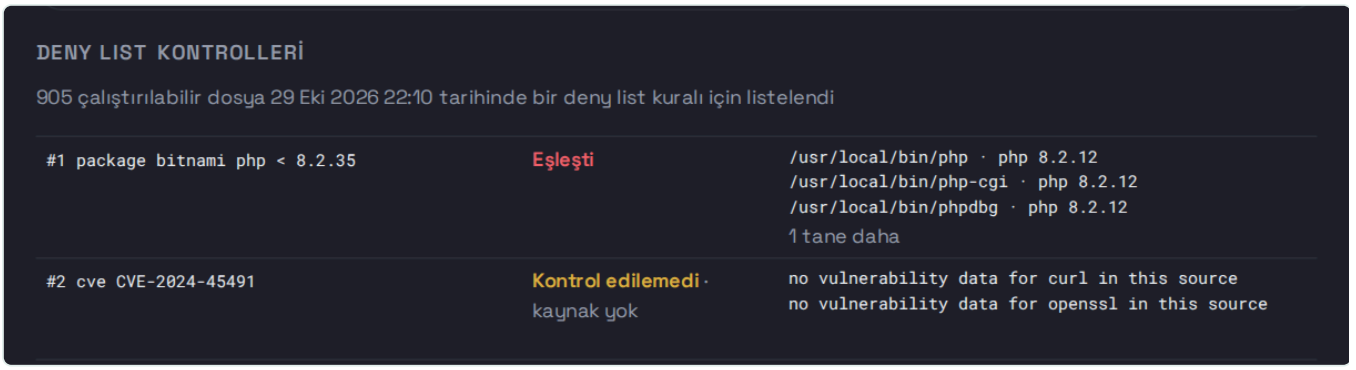
Trivy paket yöneticilerinin kaydettiğini okur. Kaynaktan derlenip image'a kopyalanmış bir runtime'ın böyle bir kaydı yoktur, Trivy onun için paket listelemez. php.net arşivinden derlenmiş bir PHP 8.2.12, etiketsiz bir image'da 91 Debian paketi gösterir ve php göstermez. Dosya kayıtlarını ve `bitnami` ekosistemini operator image'ındaki ikinci bir araç cevaplar: her image'ın çalıştırılabilir dosyalarını okuyan `attestkeep-binscan`. Scanner onu Trivy'yi çalıştırdığı gibi çalıştırır.

- **Yalnız gerektiğinde.** Listede onun cevaplayacağı bir kayıt varken çalışır: dosya hash'i ya da yolu, `bitnami` paketi ya da advisory id. Hiçbiri yoksa hiç çalışmaz ve hiçbir image ikinci kez çekilmez.
- **Yalnız baytlar.** Image, workload'un kendi pull secret'larıyla digest üzerinden çekilir, katmanlar silinen dosyalar düşülerek yukarıdan aşağı okunur ve her ELF dosyası hash'lenir. Image'ın adına, label'larına, environment'ına ve base image'ına hiç bakılmaz.
- **Sürüm dosyanın kendisinden.** Runtime'ın sürümü çalıştırılabilir dosyanın salt okunur verisinden okunur ve aynı dosyadaki ikinci bir metinle doğrulanır. Ürünler php, node, python, ruby, nginx, apache, postgresql, curl ve openssl. Strip edilmiş build de okunur: `strip` sembolleri siler, salt okunur veriyi silmez.
- **Üç cevap.** Eşleşti: dosya, sha256'sı, sürüm ve onu bulan imza ile. Eşleşmedi. Ya da nedeniyle kontrol edilemedi: kimlik bilgisi, registry, image yok, sınır, araç hatası ya da kaynak yok. Kontrol edilemedi hiçbir zaman temiz sayılmaz.

Okuma sınırları. 256 MiB'tan büyük ELF dosyası hash'lenir ama sürüm için okunmaz, image başına en fazla 4.000 ELF dosyası okunur, bir dosya taraması 60 saniyede durur. Bu sınırlar chart'taki `binscanLimits` değeridir. Çekme, tarama ve sürüm eşleştirmeyle bütün bir kontrolün kendi süresi vardır: `scan.binscanTimeout` (10 dakika). Kontrol indirmeden önce image katmanlarının boyutunu toplar, disk bütçesinden büyük image indirilmez. Bütçe, scanner'ın `/tmp` sınırı `scan.tmpSizeLimit`'in (4Gi) worker'lar arasında paylaşılmış halidir, ya da verilmişse `scan.binscanDiskBudget`. Bir sınıra ya da süreye takılan cevap *kontrol edilemedi (sınır)* olarak yazılır ve sınırın adını verir. Yönetici kontrol edilemeyen image için *Yeniden kontrol*'e basabilir, geçerli sınırlarla yeniden okunur.

Ne zaman çalışır. Yeni kayıt önce çalışan pod'ların image'larıyla, sonra son 90 günde taranan digest'lerle kontrol edilir. Yeni image taramasında kontrol edilir, kaydın henüz ulaşmadığı çalışan image'lar saatte bir kontrol edilir. Dosya ya da `bitnami` kaydı için image'ı henüz cevap almamış pod, taranmamış image gibi değerlendirilir: `DenyUntilScanned` reddeder, `AllowAndScan` uyarıyla kabul eder. Advisory kaydı beklemez: Trivy'nin bulguları hemen karar verir, kontrolün sonradan bulduğu eşleşme yalnız bunlara eklenir.

Advisory id bu runtime'lara da ulaşır. Listede advisory kaydı varken bulunan runtime'lar, operator'ün zaten tuttuğu Trivy veritabanındaki Bitnami zafiyet aralıklarıyla karşılaştırılır. Bu aralıkların kapsamadığı curl ya da openssl gibi bir runtime "no vulnerability data in this source" olarak yazılır, asla "eşleşmedi" olarak değil.



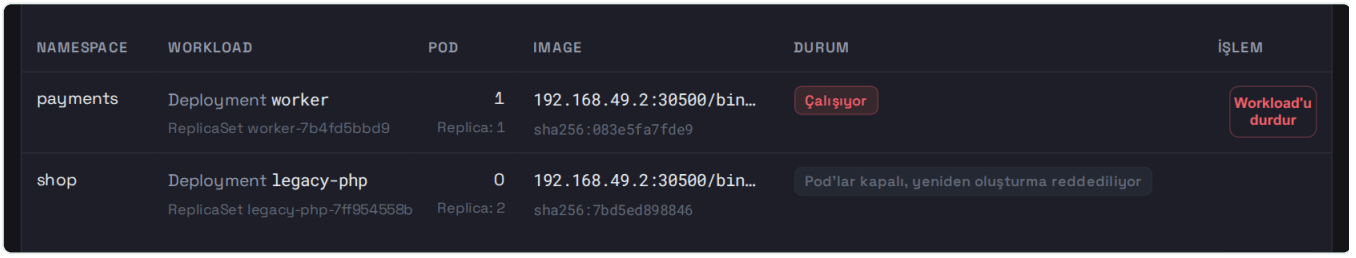
`php:8.2.12-cli` detayında *Deny list kontrolleri*, 905 çalıştırılabilir dosyanın listesinden. Kayıt 1, `bitnami php < 8.2.35`, `/usr/local/bin/php` ve 8.2.12 sürümündeki üç dosyayla daha eşleşti. Advisory id olan kayıt 2 *Kontrol edilemedi* diyor: kaynakta bulduğu curl ve openssl için zafiyet verisi yok.

Göremedikleri: sürüm metni silinmiş ya da değiştirilmiş dosya, section header'ları silinmiş dosya, yalnız sembol adıyla tanınan statik bağlanmış ve strip edilmiş dosya, paketlenmiş (packed) dosyalar, container açılırken indirilen runtime'lar, dosyanın dışında yazılmış sürümler, yorumlanan uygulama kodu ve çekilemeyen image'lar. Section header'ı olmayan dosya, paketlenmiş dosya ve çekilemeyen image nedeniyle birlikte kontrol edilemedi olarak yazılır.

Etkilenen workload'lar ve take down

Kayıt yeni pod'ları reddeder. Zaten çalışanı durdurmaz, hiçbir policy modu da bunu kendiliğinden yapmaz. Kaydın satırındaki *Etkilenen*, kaydın şu anda yakaladığı ve reddettiği workload'ları cluster'dan canlı okuyup her birini durumuyla listeler:

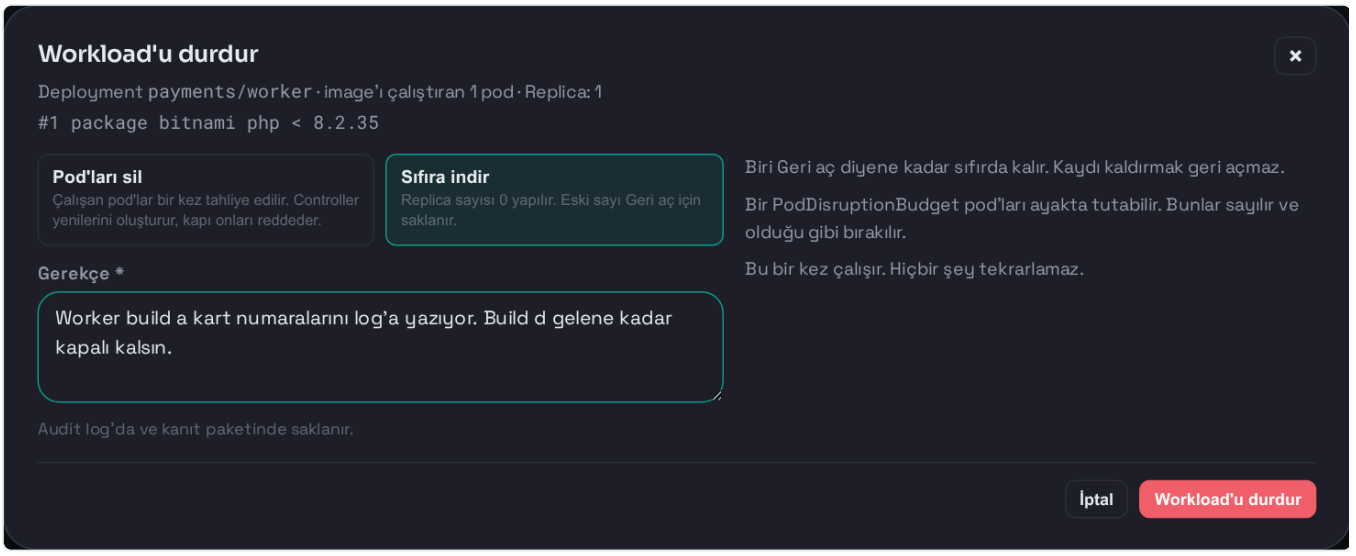
- **Çalışıyor**: listedeki image'ı çalıştıran pod'lar ayakta.
- **Pod'lar kapalı, yeniden oluşturma reddediliyor**: hiçbiri ayakta değil ve controller'ın açtığı yeni pod'lar reddediliyor.
- **Sıfıra indirildi**: bir take down replica sayısını sıfıra indirdi ve workload hâlâ orada.
- **Dışarıdan geri açıldı**: burada sıfıra indirilmişti ve yeniden replica istiyor. Biri ya da bir GitOps aracı geri koydu. Gösterilir ve öyle bırakılır.



Cluster'dan canlı okunan *Etkilenen* listesi. `payments`'taki `worker` Deployment'ı listedeki bir image'ı hâlâ çalıştırıyor, bu yüzden *Workload'u durdur* sunuluyor. `shop`'taki `legacy-php`'nin ayakta pod'u yok ve yeni pod'ları reddediliyor.

Chart'ta `rbac.denyList.takedown: true` varken yönetici çalışan bir workload'u gerekçe yazarak take down edebilir. Konsoldaki düğme *Workload'u durdur*'dur. Bir kez çalışır ve hiçbir şey onu tekrarlamaz. Take down cluster'a dokunmadan önce kaydedilir, sonucu da sonra: tamamlandı, bir budget engelledi ya da hatasıyla başarısız oldu.

- **Pod'ları sil** (varsayılan), listedeki image'ı çalıştıran pod'ları bir kez tahliye eder. İstenen durum değişmez. Controller yerine yenilerini açar, kayıt durdukça admission onları reddeder. Image değişene ya da kayıt kaldırılana kadar workload degraded görünür, sonra kendiliğinden düzelir.
- **Sıfıra indir**, Deployment, StatefulSet ya da ReplicaSet için replica sayısını scale alt kaynağıyla 0 yapar ve önceki sayıyı saklar. Sayı pencerenin gösterdiğinden farklılaştıysa reddeder. **Geri aç** sayıyı geri koyar. Kaydı kaldırmak geri açmaz: hiçbir şey bir tıklama olmadan workload'u ölçeklemez. DaemonSet, Job, CronJob ya da sahipsiz pod, pod'ları silinerek take down edilir.

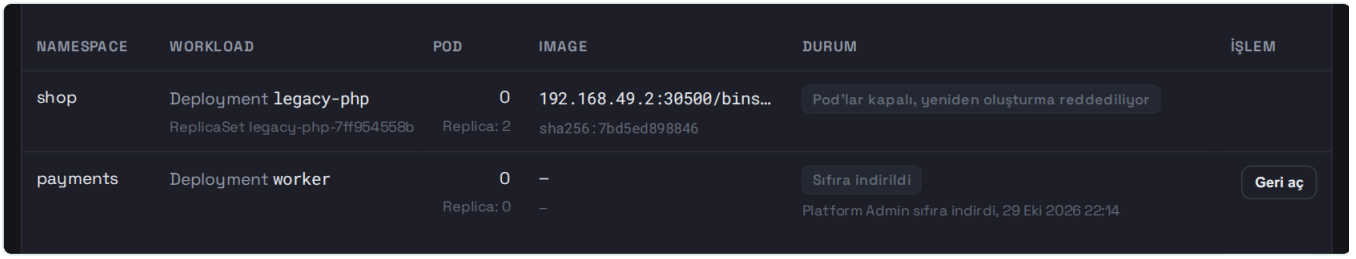


Bir pod'u çalışan `payments/worker` Deployment'ı için take down. *Sıfıra indir* seçili, böylece replika sayısı geri açmak için saklanır. Notlar, kaydı kaldırmanın workload'u geri açmadığını, bir PodDisruptionBudget'ın pod'ları ayakta tutabileceğini ve işlemin bir kez çalışıp tekrarlanmadığını söyler. Gerekçe denetim kaydında ve kanıt paketinde tutulur.

PodDisruptionBudget'lara uyulur. Pod'lar eviction API'siyle tahliye edilir, bir budget'ın koruduğu pod çalışmaya devam eder. Satır o zaman tuttuğu pod sayısıyla "PodDisruptionBudget engelledi" yazar ve hiçbir şey zorlanmaz, tekrar denenmez.

GitOps. Argo CD ya da Flux'ın izlediği workload panelde işaretlenir, pencere aracın sıfıra indirmeyi geri alacağı uyarısını verir. Geri aldığında satır *Dışarıdan geri açıldı* yazar ve Attestkeep onu bir daha indirmez. Kalıcı çözüm kaynaktadır: image'ı değiştirin ya da replica sayısını düşürün.

`rbac.denyList.takedown: true` yoksa düğme kapalı görünür ve API `409` cevabı verir. Admission'ın incelemediği bir namespace, operator'ün kendi namespace'i ya da webhook'un atladığı bir namespace, hiçbir zaman take down edilmez: yeniden açılan pod'ları kontrol edilmeyeceği için take down yalnız aynı image'ı yeniden başlatırdı.



Take down'dan sonra satır *Sıfıra indirildi* der, kimin ne zaman yaptığını yazar ve *Geri aç*'ı sunar. Geri açma saklanan replika sayısını geri koyar. Workload'u bundan başka hiçbir şey büyütmez.

Bildirim, event ve kanıt

Ret, kaydı adlandıran ve `DenyListRefused` gerekçeli bir Kubernetes Warning event'i bırakır, workload'un üstüne. Böylece `kubectl describe` pod'ların neden başlamadığını söyler. Sahipsiz pod için event namespace'e yazılır. Workload ve kayıt başına beş dakikada en fazla bir event oluşur. Reddedilen image'ın ImageSecurityReport'u, kaydı ve reddi `status.denyList` altında taşır. *Deny list eşleşmesi*'ne abone kanallar 5. bölümde anlatıldığı gibi haber alır.

Kanıt paketinde bir *Deny list* bölümü vardır: dönem sonunda yürürlükteki kayıtlar ve dönemde eklenen ya da kaldırılanlar, kim, ne zaman ve neden; kayıtların reddettiği ve içeri aldığı admission'lar; istisnalar, yani namespace listeyi denetlediği için içeri alınan eşleşmeler; gerekçesiyle her take down ve geri açma; dosya kaydı için de dosya, hash'i ve sürüm. Policy hash'i listeyi kapsamaz, bir admission'a karar veren kayıt o admission'ın satırında yazar.

Kim ne yapabilir

Listeyi, eşleşmelerini ve etkilenen workload'ları okumak ve What if çalıştırmak, policy okumakla aynı yetkiyi ister. Kayıt eklemek ve kaldırmak, workload'u take down etmek ve geri açmak yönetici işidir, geri açma dışındakiler gerekçe ister. Yeniden kontrol de yönetici işidir. Audit log bunları `deny_list_add`, `deny_list_remove`, `deny_list_takedown`, `deny_list_restore` ve `deny_list_check_again` olarak yazar. API `/api/v1/deny-list` altındadır. Sunucunun reddettiği alan, alanın adıyla `400` olarak döner. Aynı değerle ikinci bir etkin kayıt `409`'dur.

8. Falco'dan gelen runtime event'leri

Falco, container'lar başladıktan sonra ne yaptıklarını izler. Attestkeep Falco'nun event'lerini alabilir, her birini geldiği image ve pod'la birlikte saklar ve konsolda yazdığınız runtime policy'lerine göre işlem yapar.

Attestkeep Falco'yu paketlemez, kurmaz, yapılandırmaz ve Falco'ya yazmaz. Tespiti sizin Falco'nuz ve onun kuralları yapar. Runtime event'i admission kararını değiştirmez. Runtime policy'sinin yapabildikleri bildirim göndermek, issue açmak ve event'in geldiği pod'u durdurmaktır.

Akış

1. Bir node'da Falco kuralı tetiklenir, örneğin biri bir container'da interaktif shell açtığında.
2. Event JSON olarak, mutual TLS ile operator'ün integrations portuna gönderilir. Falco bunu kendi `http_output`'undan gönderir ya da Falcosidekick webhook çıkışından iletir. Her iki durumda da gönderen, operator'ün verdiği istemci sertifikasını sunar.
3. Operator event'i saklar ve container'ın çalıştırdığı image ve digest'e bağlar.
4. Runtime policy'leri sırayla denenir. Eşleşen ilk açık policy ne olacağına karar verir. Hiçbiri eşleşmezse event kayıt olarak tutulur, hiçbir şey gönderilmez.
5. Event ve karar Falco > Event'ler sayfasında, Images sayfasında image'ın yanında ve kanıt paketinde görünür.

Gerekenler

- Upstream `falcosecurity/falco` Helm chart'ıyla kurulmuş Falco. Dokümantasyondaki değerler chart 9.2.0 (Falco 0.45.0, Falcosidekick 2.32.0) içindir. Eski sürümler kontrol edilmedi.
- `falco.json_output: true`.
- Falco ya da Falcosidekick pod'larından operator Service'inin integrations portuna (varsayılan 8444) ağ erişimi.
- Admin hesabı. Entegrasyon sayfasını, sertifika indirmeyi ve policy düzenlemeyi yalnız admin'ler görür.
- Pod durduran policy için Attestkeep chart'ında `rbac.allowEnforce: true`. `rbac.denyList.takedown` bunun yerine geçmez.
- Mesaj ve issue için Runtime uyarısına abone bir bildirim kanalı ya da GitHub, GitLab veya Jira kanalı.

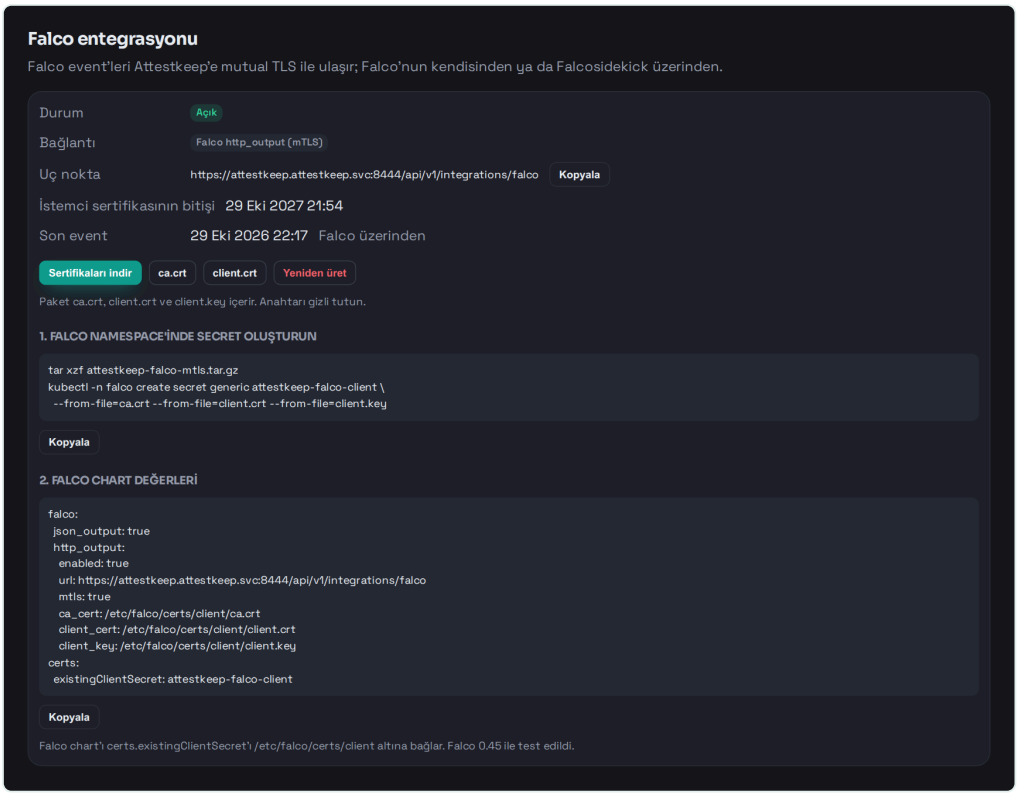
Falco'yu bağlamak

Entegrasyonu Attestkeep chart'ında açın ve bağlantı yolunu seçin:

```
integrations:
  port: 8444
  falco:
    enabled: true
    connection: mtls   # ya da sidekick
```

Varsayılan `mtls`'te Falco her event'i kendi `http_output`'undan doğrudan operator'e gönderir. Falco'nuz zaten Falcosidekick'e gönderiyorsa `sidekick` seçin. O zaman Sidekick'in webhook çıkışı aynı istemci sertifikasıyla operator'e iletir. Başka bir değer chart'ı hata verdirir.

Entegrasyon açıldığında operator kendi küçük CA'sını ve bir istemci sertifikası (`CN=falco`, bir yıl geçerli) üretip `<release>-falco-mtls` Secret'ında tutar. Integrations portu tek bir yol sunar, `POST /api/v1/integrations/falco`, ve yalnız bu CA'nın imzaladığı istemci sertifikasını kabul eder. Token yoktur.

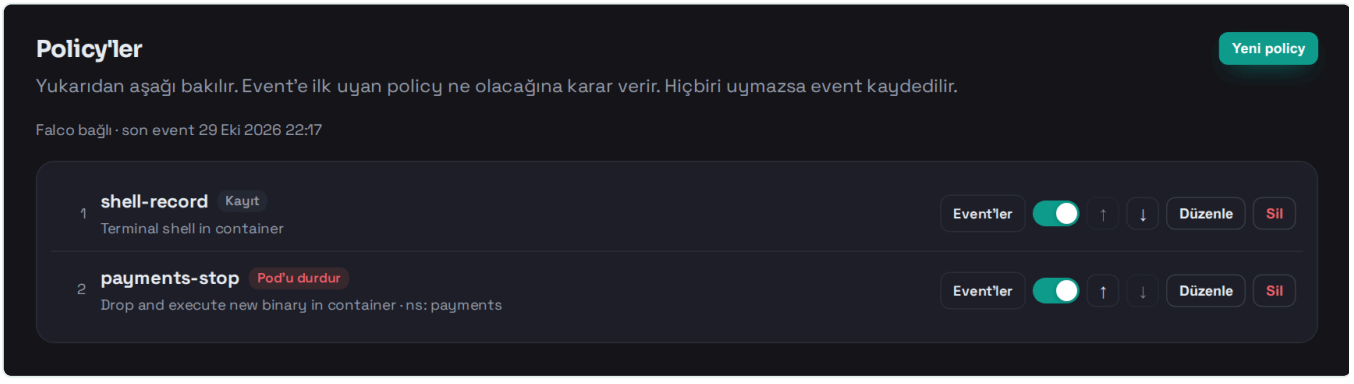


Bu kitapta kullanılan kurulumda Falco > Falco entegrasyonu. Durum açık, bağlantı Falco'nun kendi `http_output`'u üzerinden mTLS, son event Falco'dan geldi. *Sertifikaları indir* `ca.crt`, `client.crt` ve `client.key` içeren bir paket kaydeder. Altında Falco namespace'inde Secret'ı oluşturan komut ve seçilen bağlantı için Falco chart değerleri var.

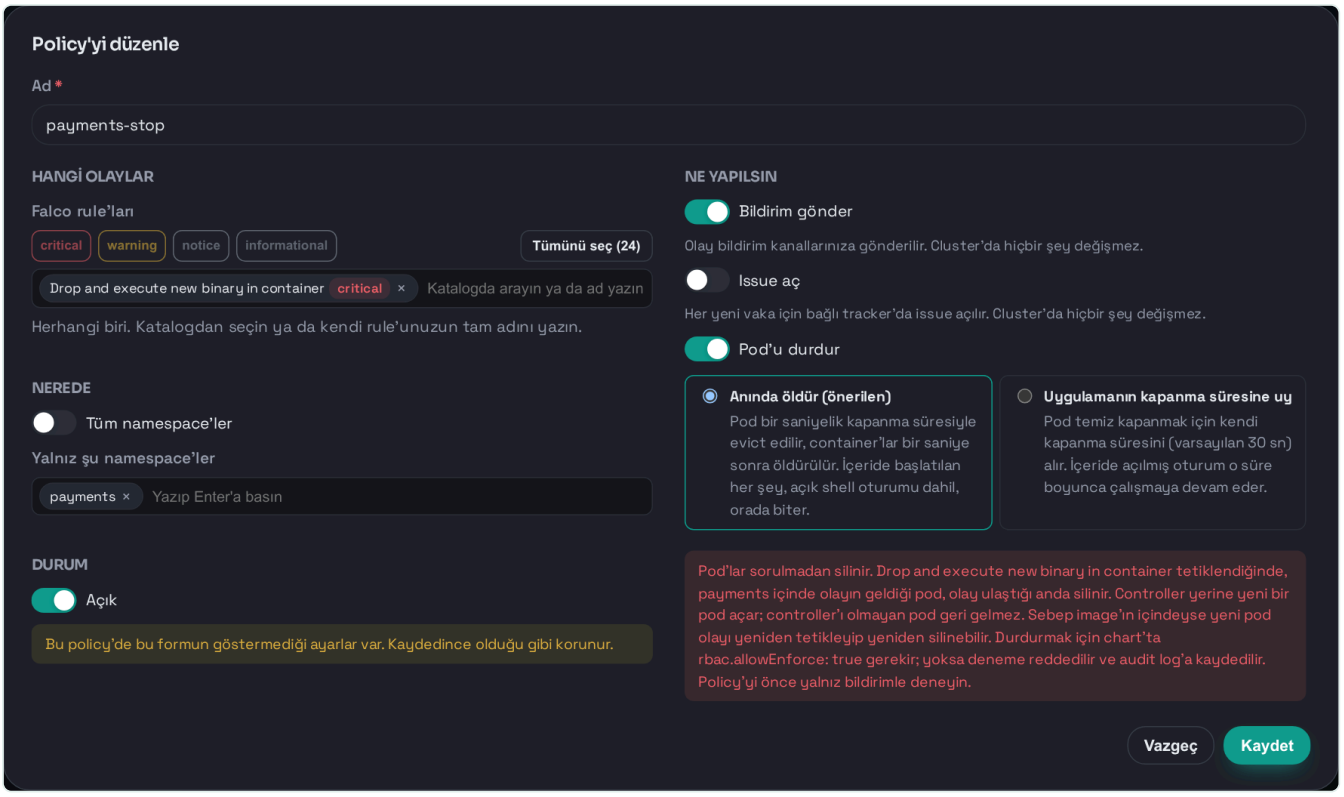
Her paket indirme audit log'a yazılır. `client.key` gizli tutulmalı, çünkü elinde bulunduran herkes policy'lerinizin işlem yapacağı event gönderebilir. *Yeniden üret* aynı CA'dan yeni bir istemci sertifikası verir. Eskisi süresi dolana kadar çalışmaya devam eder, Secret'ı değiştirirken event akışı kesilmez. Operator'ün kendi sunucu sertifikası yenilendiğinde, paket yeniden indirilene kadar sayfada "Sertifikalar <tarih> tarihinde değişti" uyarısı görünür.

Runtime policy'leri

Falco > Policy'ler runtime policy'lerinizi sırasıyla listeler. Saklanan her event yukarıdan aşağıya denenir, eşleşen ilk açık policy karar verir. Sıra oklarla değişir. Değişiklik kaydeden replica'da hemen, diğer replica'larda 15 saniye içinde geçerli olur.



Yukarıdan aşağı kontrol edilen iki runtime policy'si. `shell-record` *Terminal shell in container* olayını kaydeder. `payments-stop` `payments`'ta *Drop and execute new binary in container* olunca pod'u durdurur. Policy burada, Event'lerde ve kanıt paketinde etkisiyle etiketlenir.



`payments-stop` formu: tek Falco rule'u, tek namespace, *Bildirim gönder* ve *Pod'u durdur* açık, *Anında öldür (önerilen)* seçili. Kırmızı kutu eşleşen pod'ların sorulmadan silineceği konusunda uyarır ve 1.4.0'dan itibaren durdurmanın chart'ta `rbac.allowEnforce: true` istediğini söyler.

Alan	Görevi
Ad	Zorunlu, en fazla 128 karakter, benzersiz.
Falco rule'ları	Bu kurallardan herhangi birinin event'iyle tam adıyla eşleşir. Rule'lar listesinden seçilir ya da kendi kuralınızın adı yazılır. En az bir tane gerekir.
Tüm namespace'ler / Yalnız şu namespace'ler	Policy'yi listedeki namespace'lerdeki pod'lardan gelen event'lerle sınırlar.
Bildirim gönder	Event'i Runtime uyarısına abone her açık kanala gönderir.
Issue aç	Runtime uyarısına abone tracker kanallarında issue açar.
Pod'u durdur	Event'in geldiği pod'u *Anında öldür* ya da *Uygulamanın kapanma süresine uy* seçeneğiyle durdurur.
Durum	Açık ya da kapalı. Yeni policy kapalı başlar. Kapalı policy saklanır, bir şey yapmaz.

Üç aksiyondan biri açıksa policy `notify` aksiyonuyla kaydedilir. Üçü de kapalıysa `record` olarak kaydedilir: eşleşen event policy'ye bağlanarak kaydedilir, başka bir şey olmaz. API (`/api/v1/runtime/rules`) formun gösterdiğinden fazla koşul kabul eder: tag, priority, source, output alanları, image'ın son kararı ve imzası, `duration` içinde `count` event'lik pencere. Ayrıca event'i saklayıp başka bir şey yapmayan `allow` aksiyonu da vardır. Formun göstermediği ayarları olan bir policy açıldığında form bunu söyler ve kaydederken bu ayarları korur.

Pod'u durdurmak

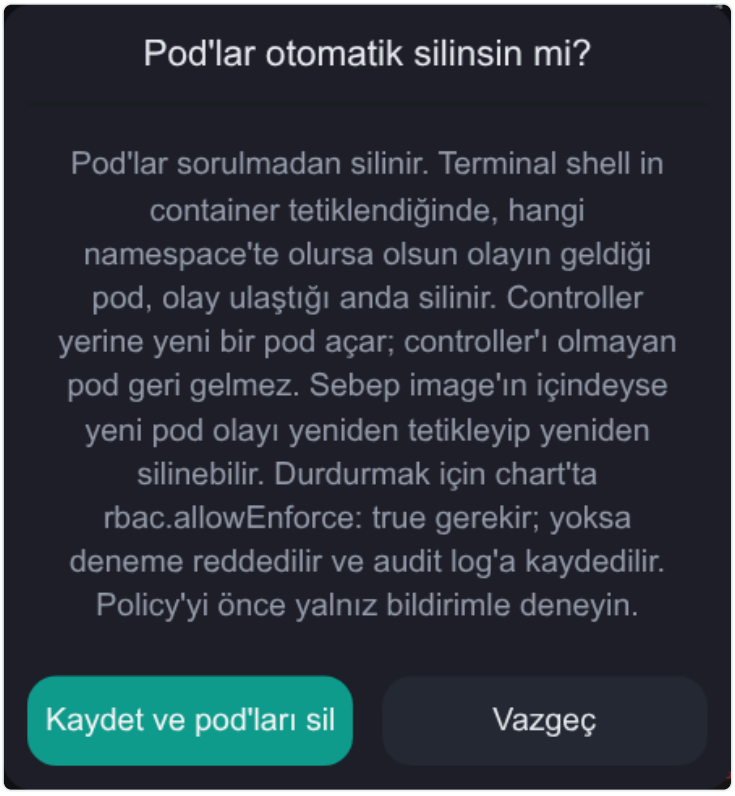
Pod'u durdur açık bir policy, event'te adı geçen pod'u event gelir gelmez durdurur. Policy'de pencere varsa pencere dolduğunda durdurur. Operator API server'dan pod'un evict edilmesini ister, bu yüzden PodDisruptionBudget'lar geçerlidir. Budget izin vermezse pod çalışmaya devam eder ve audit log nedenini yazar.

- *Anında öldür* (`stop_mode: immediate`, varsayılan): eviction bir saniyelik kapanma süresiyle yapılır. Container içinde başlatılan her şey, açık bir shell de dahil, orada biter.
- *Uygulamanın kapanma süresine uy* (`stop_mode: graceful`): pod kendi termination grace period'unu alır, pod başka bir değer vermediyse 30 saniye. İçeride açılmış shell bu süre boyunca çalışır.

Pod'un controller'ı yeni bir pod başlatır. Controller'ı olmayan pod geri gelmez. Sebep image'ın içindeyse yeni pod aynı event'i tetikleyip yeniden durdurulabilir.

- Durdurmak için, ClusterRole başka ne verirse versin, `rbac.allowEnforce: true` gerekir. 1.4.0'dan itibaren `rbac.denyList.takedown: true` take down için pod eviction yetkisi vermiş olsa bile bu kural geçerlidir. Yetki yoksa durdurma API server'a sorulmadan reddedilir, audit log "not stopped: the chart has not granted pod eviction (set rbac.allowEnforce)" yazar ve uyarı pod'un hâlâ çalıştığını söyler. `runtimeReconciliation: enforce` taraması da aynı kurala uyar.
- Operator'ün kendi namespace'indeki pod'lar hiçbir zaman durdurulmaz. Başka namespace dışarıda tutulmaz, durdurma policy'sini sistem namespace'lerinden uzak tutmak için *Yalnız şu namespace'ler* ayarını kullanın.
- Event'in pod'u adlandırması gerekir (`k8s.ns.name` ve `k8s.pod.name`). Falco chart'ının varsayılanlarıyla bu alanlar gelir.
- Zaten kapanmakta olan pod'a dokunulmaz. Her operator replica'sı aynı pod için en fazla 10 dakikada bir durdurma ister.

Pod'u durdur açık bir policy kaydedilmeden önce sorulur. Pencere pod'ların sorulmadan silineceğini, controller'ı olan ve olmayan pod'a ne olacağını ve durdurmanın chart'ta `rbac.allowEnforce: true` istediğini yazar.



Pod'u durdur yeni açılmış bir policy kaydedilirken sorulan soru. Rule'u ve namespace'leri adlandırır, controller'ı olan ve olmayan pod'a ne olacağını söyler ve durdurmanın `rbac.allowEnforce: true` istediğini yazar. Bu ayar yoksa deneme reddedilir ve denetim kaydına yazılır.

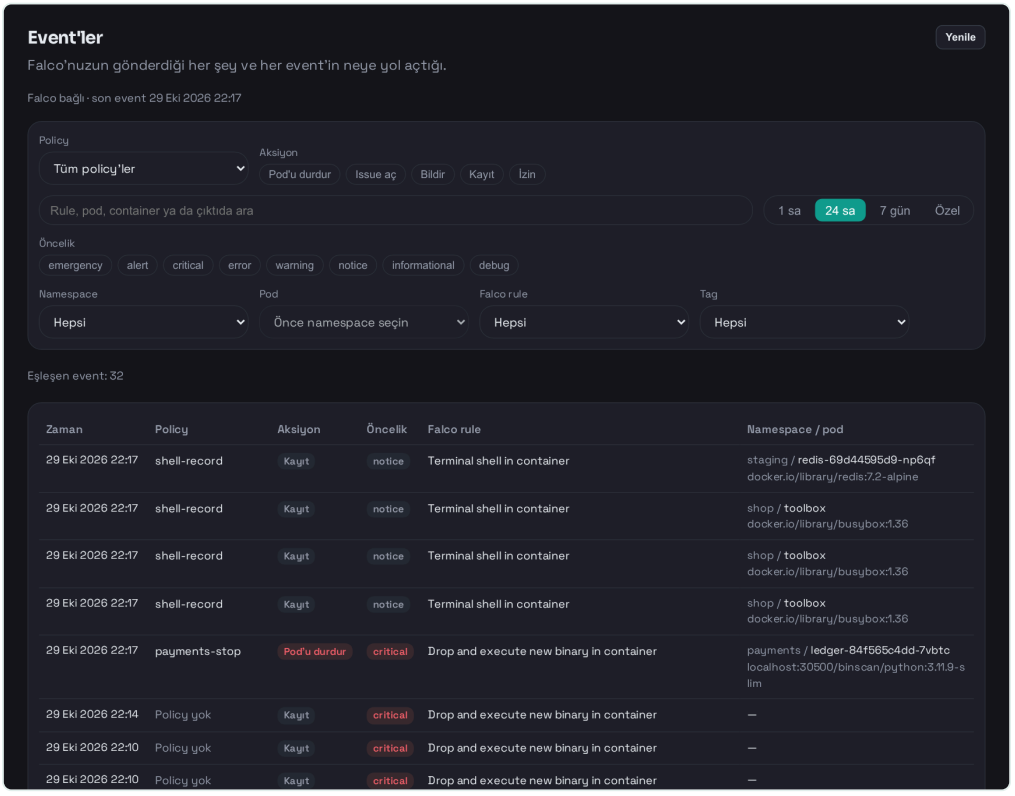
Her deneme audit log'a `runtime_pod_stop` olarak yazılır. Aktör `policy:<ad>`, hedef pod'dur. Ayrıntıda Falco kuralı, pod UID, container, image, digest, node, event id ve sonuç bulunur. Pod'u durdurmak admission'ı değiştirmez, aynı image'ın yeni pod'ları eskisi gibi kabul edilir.

Bildirim ve issue

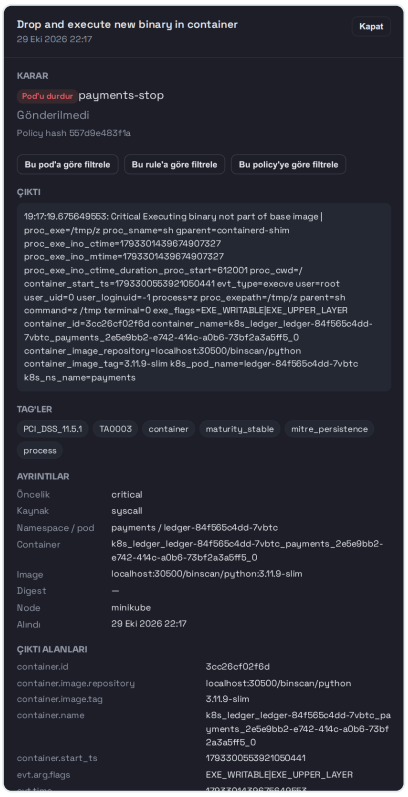
Mesajın başlığı `Runtime alert: <Falco kuralı>` olur. Gövdede Falco'nun output satırı, kural, priority, source, namespace, pod, container, image, digest, node, zaman, policy ve aksiyonu bulunur. Policy pod'u durdurduysa mesaj bunu yazar. Bir policy image digest başına saatte en fazla bir mesaj gönderir. Bu sınır mesaj ve issue için geçerlidir, pod'u durdurmayı geciktirmez.

Issue yalnız event bir image digest'ine bağlanabildiğinde açılır: digest ve Falco kuralı başına bir issue, `RUNTIME: <kural> in <image>` başlığıyla ve `runtime` etiketiyle. O image'da kural 7 gün tetiklenmezse `quiet`, digest'i 24 saattir hiçbir çalışan pod kullanmıyorsa `image_retired` olarak kendiliğinden kapanır. İkisi de sorunun düzeltildiği anlamına gelmez. Event, ancak onu alacak en az bir açık kanal varsa bildirilmiş sayılır.

Event'ler



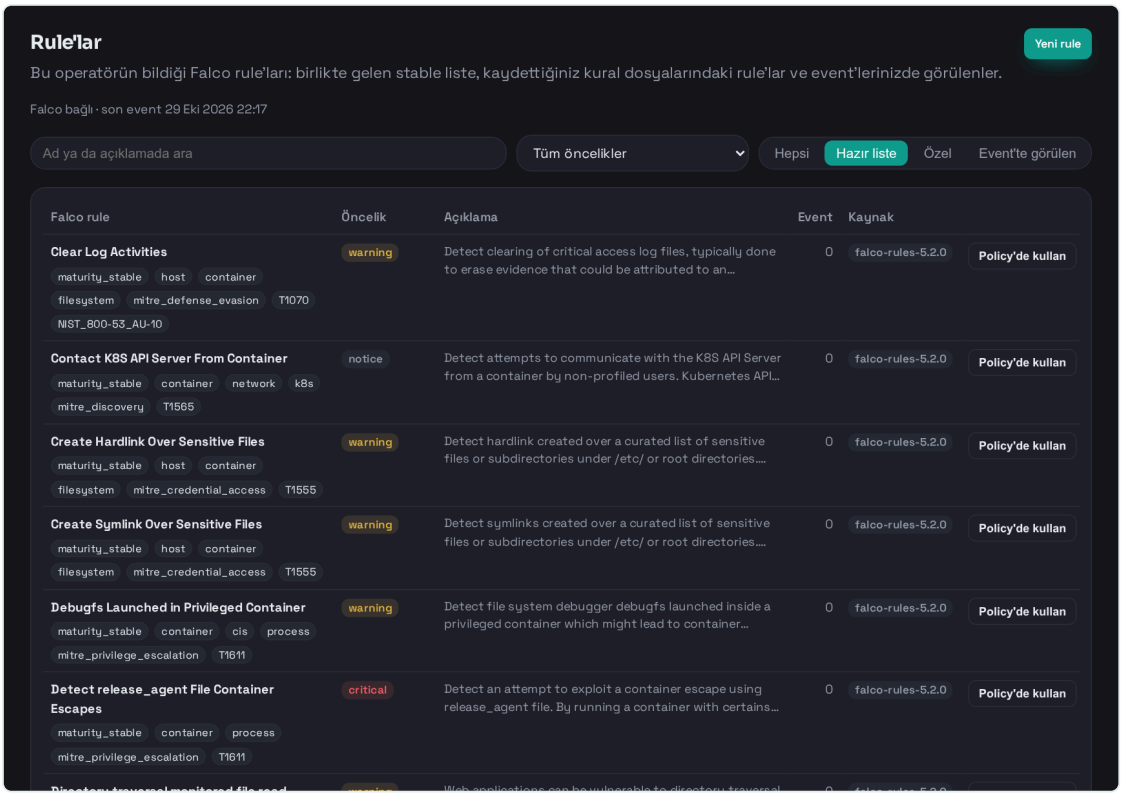
Son 24 saatte Falco > Event'ler: 32 event. Her satır eşleşen policy'yi ve eylemini, önceliği, Falco rule'unu, namespace'i ve pod'u verir. Hiçbir policy'nin eşleşmediği event'ler *Policy yok* diye görünür ve kayıt olarak tutulur. Filtreler policy, eylem, serbest metin, öncelik, namespace, pod, Falco rule'u ve etiketi bir zaman aralığında süzer.



Açılmış bir event: `payments`'ta *Drop and execute new binary in container*, `payments-stop` tarafından *Pod'u durdur* kararı verilmiş. Panel kararı, mesajın gönderilip gönderilmediğini, policy hash'ini, Falco'nun çıktı satırını, etiketleri ve event'in bağlandığı namespace, pod, container, image ve node'u gösterir. Bu kurulumda chart `rbac.allowEnforce` ayarlamıyor, bu yüzden durdurma reddedildi ve pod çalışmaya devam etti.

Falco kuralları ve rule editörü (beta)

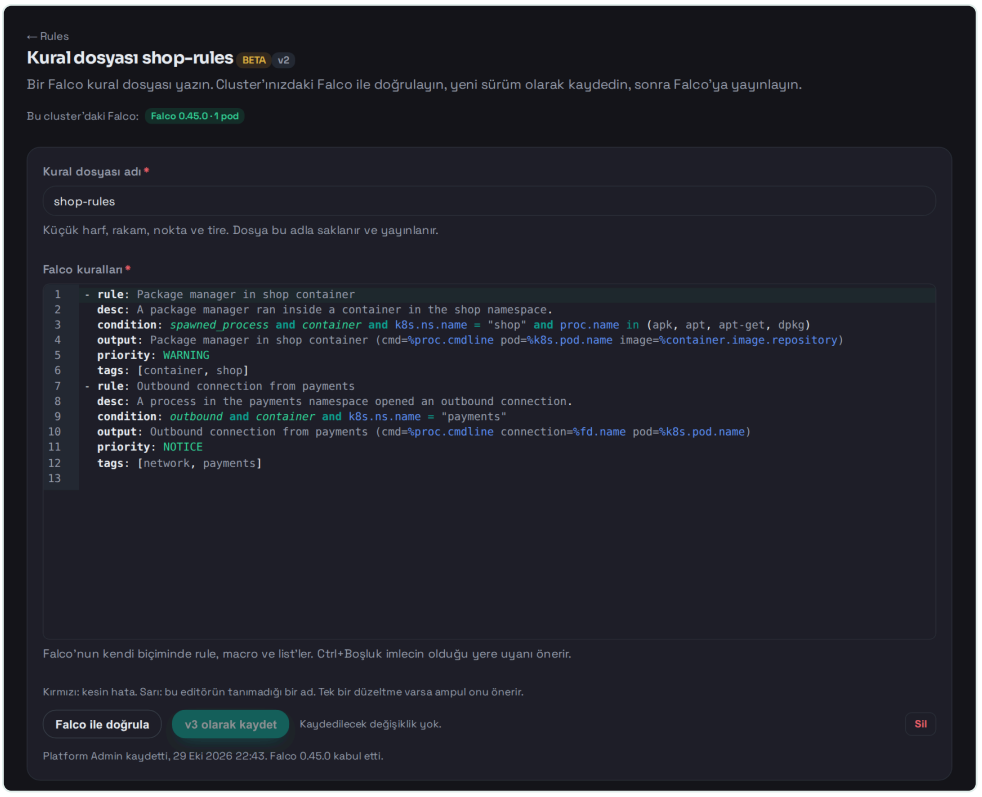
Falco > Rule'lar operator'ün bildiği Falco kurallarını listeler: ürünle gelen `falcosecurity/rules` stable listesindeki 25 kural, kaydettiğiniz kural dosyalarındaki kurallar ve event'lerde görülen kural adları. *Policy'de kullan* bir kuraldan policy başlatır.



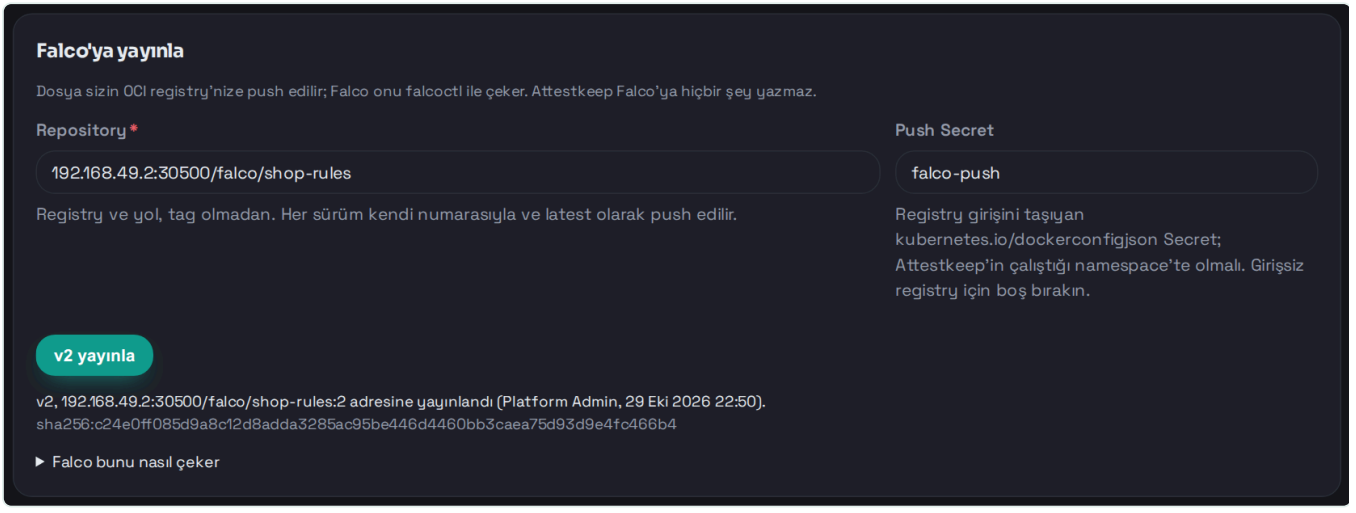
Hazır liste filtresiyle Falco > Rule'lar. Her rule önceliğini, etiketlerini, açıklamasını, burada kaç event ürettiğini ve kaynağını, `falco-rules-5.2.0`'ı gösterir. *Policy'de kullan* rule'dan bir runtime policy'si başlatır.

Yeni rule (admin) bir Falco kural dosyası için editör açar: Falco'nun kendi YAML'ıyla rule, macro, list ve override'lar. Editör beta aşamasında. Tamamlama ve kontroller cluster'da çalışan Falco sürümüne göre yapılır. Falco, `app.kubernetes.io/name=falco` etiketiyle bulunur.

- *Falco ile doğrula*, operator'ün namespace'inde Falco'nuzun kullandığı image'la tek seferlik bir pod başlatır. Pod Falco'nun varsayılan kurallarını ve sizin dosyanızı yükler, Falco'yu doğrulama modunda çalıştırır ve sonuç okununca silinir. Çalışan Falco'nuza dokunulmaz.
- Kaydet, ancak Falco editördeki metnin tam halini kabul ettiyse açılır. Sunucu kaydetmeden önce yeniden doğrular. Her kayıt yeni bir sürümdür, kimin kaydettiği ve hangi Falco sürümünün kabul ettiği yazılır. Başka kayıtlı dosyada tanımlı bir kural adı reddedilir.
- Yayınlama, kayıtlı bir sürümü kendi OCI registry'nize falcoctl `rulesfile` artifact'i olarak, sürüm numarası ve `latest` tag'leriyle push eder. Registry giriş istiyorsa operator'ün namespace'indeki bir `kubernetes.io/dockerconfigjson` Secret'ının adını verirsiniz. Attestkeep Secret'ı push sırasında okur, kimlik bilgisini saklamaz.



Sürüm 2'deki kayıtlı `shop-rules` kural dosyası. Başlığın altındaki satır cluster'da bulunan Falco'yu gösterir: 1 pod'da Falco 0.45.0. v3 olarak kaydetme, metin değişip Falco kabul edene kadar kapalı kalır. Son satır sürüm 2'yi kimin kaydettiğini ve hangi Falco sürümünün kabul ettiğini söyler.



Falco'ya yayınla: kendi registry'nizdeki repo, push Secret'ının adı ve sürüm 2'nin digest'iyle yayımlandığı kayıt. *Falco bunu nasıl çeker* Falco chart'ınız için falcoctl değerlerini açar.

Falco dosyayı falcoctl ile yükler. Falco chart değerlerinde kendi referansınızı hem `install.refs` hem `follow.refs` listesine ekleyin ve chart'ın varsayılan referansı `falco-rules:5` 'i silmeyin. Helm listeleri birleştirmez, değiştirir. Bu referans olmazsa falcoctl `/etc/falco/falco_rules.yaml` dosyasını kurmaz. Sonra dosyanızı `rules_files` listesine Falco'nun kendi kurallarından sonra ekleyin. Konsolda bir dosyayı silmek kurallarını Rule'lar listesinden kaldırır. Daha önce yayınlanan dosya, falcoctl'den referansı kaldırana kadar registry'de ve onu kurmuş her Falco'da kalır.

Saklama ve metrikler

Her event saklanır, Attestkeep'in kendi priority eşiği yoktur. `runtime.retentionDays` (varsayılan 90) günden eski event'ler saatte bir silinir, `0` süresiz tutar. Issue geçmişi, audit kayıtları ve policy revizyonları silinmez. Hiç saklamak istemediğiniz event'leri Falco'nun kendi `falco.priority` ayarıyla kesin.

```
attestkeep_runtime_events_total{source, priority}
attestkeep_runtime_events_dropped_total{reason="duplicate|unresolved_image|throttled"}
attestkeep_runtime_alerts_notified_total
attestkeep_runtime_rule_matches_total{rule, action}
```

`attestkeep_runtime_rule_matches_total` içindeki `rule` runtime policy'sinin adıdır. `unresolved_image` event'in kaybolduğu anlamına gelmez. Falco host üzerindeki process'leri de izler, böyle bir event image'sız saklanır.

9. Enforcement'ın çalıştığını görmek

Çalışmayan bir admission webhook dışarıdan kurallara uyan bir cluster gibi görünür. Bu kontrol beş dakika kadar sürer, yalnız kubectl kullanır ve bir reddi kendi gözünüzle görerek biter. Kurulumdan ve yükseltmeden sonra, ayrıca enforcement'tan emin olmak istediğiniz her seferde çalıştırın.

1. Policy'nin reddetmesi gereken bir pod deneyin

```
kubectl run enforcement-check --image=nginx:latest
```

API server webhook'u aktararak reddeder. Varsayılan policy ile çalışan bir kurulumun çıktısı:

```
Error from server: admission webhook "image-security.attestkeep.com" denied the request:
image security policy: nginx:latest: the :latest tag is blocked by policy
```

Mesaj *Error from server* ile başlıyor, yani ret API server'ın admission zincirinden geliyor. Gerekçe policy kuralını yazıyor, CI log'unda okuyan geliştirici neyi değiştireceğini bilir.

2. Pod'un oluşmadığını kontrol edin

```
kubectl get pod enforcement-check
Error from server (NotFound): pods "enforcement-check" not found
```

3. Kararın kaydedildiğini kontrol edin

Paneldeki *Son engellenen kabuller* tablosu artık bu retle ve aynı gerekçeyle başlar. `attestkeep_admission_decisions_total{decision="denied"}` metriği artmıştır.

4. Pod kabul edildiyse

- Namespace denetim dışıdır: kube-system, cert-manager, operator'ün kendi namespace'i ya da `attestkeep.com/webhook=bypass` etiketli bir namespace.
- Policy denediğiniz şeyi kapsamıyordur. `enforcement: Enforce` ve `blockLatestTag: true` değerlerine bakın.
- Webhook açık hata modundadır. `attestkeep_webhook_distrusted` değerinin 0 olduğunu ve pod'ların hazır olduğunu kontrol edin.
- Kurulum aktive edilmemiştir. Anahtar girilene kadar hiçbir şey enforce edilmez ve her admission cevabı bunu uyarı olarak yazar.

DenyUntilScanned ile

`admissionTiming: DenyUntilScanned` altında yeni bir image, taraması oluşana kadar reddedilir. Bir Community kurulumunun çıktısı:

```
kubectl run cold-check --image=busybox:1.36.1 --restart=Never -- sleep 3600
Error from server: admission webhook "image-security.attestkeep.com" denied the request:
image security policy: busybox:1.36.1: image has not been scanned yet; a scan has been queued — retry
shortly

# biraz sonra, aynı komut:
pod/cold-check created
```

10. Kanıt paketleri ve güven modeli

Kanıt paketi her denetim dönemi için imzalı tek bir belgedir. Cluster içinde, sizin kayıtlarınızdan üretilir ve denetçi kendi araçlarıyla doğrulayabilir. Denetim için kanıttır, sertifika değildir. Belgenin üstünde de bu yazar.

Eşlenen kontrol setleri

Düzenleme	Eşlenen
SOC 2, AICPA Trust Services Criteria	CC6–CC9 ve Availability A1'den 13 kriter.
EU CRA ve NIS2: (EU) 2024/2847 ve Madde 21(2)	İki düzenlemeden toplam 34 gereksinim.
DORA, Tüzük (EU) 2022/2554	Madde 8–11 ve Madde 28'den 22 gereksinim.
NIST SP 800-190	Image, registry, orchestrator ve container için 15 önlem.
SSDF, NIST SP 800-218 v1.1	PO, PS, PW ve RV'den 13 pratik.
NIST SP 800-53 Rev. 5	CM, RA, SI, SR, AU ve CP'den seçilmiş 15 kontrol.
ISO/IEC 27001:2022 Ek A	12 madde: 10 Ek A kontrolü ve kapsam dışı işaretlenen maddeler.
GDPR Madde 32	İşleme güvenliğine dair 6 madde.
PCI DSS v4.0.1 (PCI SSC, Haziran 2024)	6.3, 6.5, 10.2 ve 11.3 gereksinimlerinden 13 madde.
MAS Technology Risk Management Guidelines (18 Ocak 2021)	3.3, 6.1, 6.3, 7.2–7.6, 11.3, 11.4, 12.1, 12.2, 13.1, 13.6 ve 15. bölümlerden 30 madde.
ASD Essential Eight Maturity Model (Kasım 2023)	46 madde: uygulama yamaları, işletim sistemi yamaları ve application control, Maturity Level 1–3.
ISMAP Control Criteria (1 Ağustos 2024 tarihli Japonca basım)	Ek tablo 3'ün kontrol kriterlerinden 42 madde.
ISMS-P Certification Criteria (PIPC ve MSIT tebliği Ek 7, KISA kılavuzu Kasım 2023)	1. ve 2. alanlardan 32 madde.

Toplam on üç düzenleme ve 293 madde, her biri kendi referansıyla. Eşleme operator'ün içinde gelir, internete çıkışı olmayan cluster da aynı raporu üretir. Her kontrol dört durumdan birine düşer: *met*, *partial*, *gap* ya da *outside_scope*. Bir madde yalnız dönemin kayıtları kontrolün çalıştığını gösteriyorsa met olur, üründe o özelliğin bulunması yetmez. Cluster'ın cevaplayamayacağı madde gerekçesiyle kapsam dışı raporlanır ve yukarıdaki sayıya dahildir.

Kaç framework için paket üretilebileceği lisansa bağlıdır (13. bölüm).

Çeviriden okunan kaynaklar. İki eşleme İngilizce yayımlanmamış metinlerden okunur ve her maddede bunu yazar. **ISMAP** için İngilizce bir referans çevirisi var. Girdinin yalnız ISO/IEC 27002'ye işaret ettiği yerlerde yükümlülük 1 Ağustos 2024 tarihli Japonca basımdan çevrilir ve referans bunu belirtir. **ISMS-P** kriterlerinin resmi İngilizce metni yok. Her madde, KISA kılavuzundaki Korece kriter cümlesinin bizim çevirimizdir. Her ISMS-P kriteri teknik bir önlemi bir prosedür, görevlendirme ya da periyodik gözden geçirmeyle birlikte ister. Cluster kayıtları yalnız teknik kısmı cevapladığı için hiçbir ISMS-P kriteri tam karşılandı olarak raporlanmaz.

Yeni eşlemeleri beş attestation besler. `patch-window-48-hours`, kritik bulgular için beyan edilen süre en fazla 48 saatse ve istismar edilen zafiyetler admission'da reddediliyorsa met olur. `patch-window-two-weeks`, yüksek bulgular için beyan edilen süre en fazla iki haftaysa met olur. `runtime-observed`, Falco entegrasyonu açıksa ve dönemde en az bir event geldiyse met olur. Entegrasyon açık ama event gelmediyse partial'dır, çünkü sessiz bir cluster ile göndermeyi bırakmış bir Falco operator'den aynı görünür. Entegrasyon kapalıysa gap'tir. `producer-statement-honoured`, policy VEX beyanlarını

uyguluyor ve en az bir issuer'a güveniyorsa met olur. `context-aware-thresholds`, escalation hem `exposed` hem `privileged` için açıksa met, ikisinden biri seçiliyse partial, `Ignore` modunda gap'tir. `remediation-windows` yalnız bir issue tracker bağlıyken met olabilir, çünkü düzeltmenin ne zaman geldiğinin kaydı ancak o zaman tutulur.

Her gereksinim nasıl cevaplandı

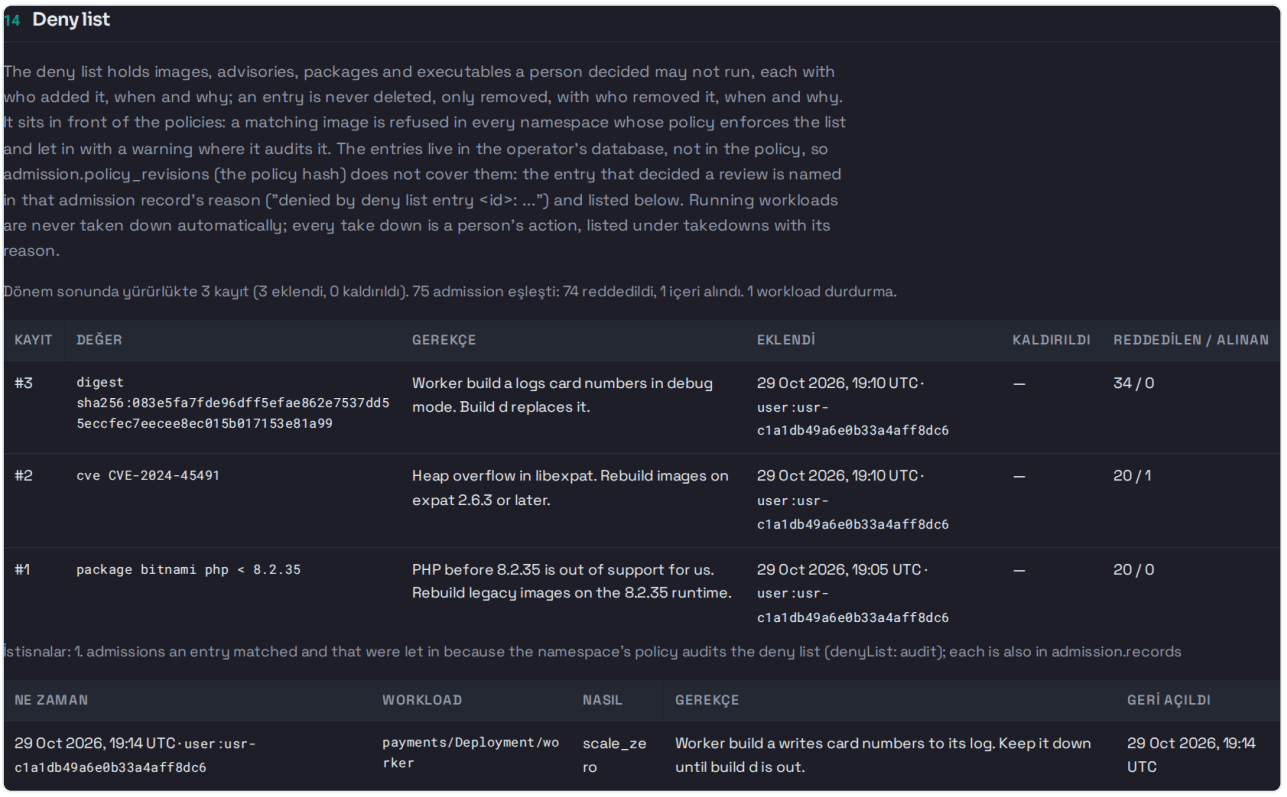
Her gereksinim onu cevaplayan testleri listeler. Testin bir yöntemi (olmuş bir şeyin kaydını okuyorsa `TEST`, bir yapılandırmayı okuyorsa `EXAMINE`), sonucu, gözlemi, rapor içindeki yolları, okuduğu kaynak ve okuma zamanı vardır. Her gereksinim OSCAL terimleriyle `result` ve `implementation_status` da taşır. PDF testleri her gereksinimin altına basar.

`generation` bloğu belgenin ne zaman ve neyle üretildiğini, kaynak sistemi, sorgu aralığını (bitiş hariç, UTC), her bölümün sorgusunu, satır sayısını ve sınırlı sorgunun sınıra takılıp takılmadığını ve kişilerin nasıl yazıldığını söyler. 1.3.0'dan itibaren `nodes {count, ceiling}` da taşır: üretim anındaki Ready node sayısı ve lisanstaki node tavanı, yalnız sayı olarak. `admission.records[]` dönemin her admission isteğini sınırsız ve satır satır taşır, denetçi örneklemini dönemin bütün kararlarından çekebilir. `organisation.declarations` operatörün zaman kaynağı ve saklama beyanlarını operatör beyanı olarak taşır.

Kişiler adla değil hesapla yazılır. `generated_by`, audit satırları için `chain[].actor`, `triage[].triaged_by` ve `runtime.audit_entries[].actor` alanları `user:<hesap id>` taşır. Runtime policy'sinin durdurduğu pod `policy:<ad>` olarak yazılır. Konsoldaki audit log adları göstermeye devam eder, denetçi sorduğunda operatör id'yi oradan çözer. 1.2.0'dan önce kaydedilmiş, yalnız ad taşıyan satırlar görünen ad ya da e-postayla birebir tek bir hesaba eşlenir, eşlenemeyen `user:unresolved` olarak yazılıp sayılır. Her admission'ı isteyen Kubernetes kimliği, `system:` kimliği ise olduğu gibi, değilse `k8s-user:` ve SHA-256'sının ilk 16 hex karakteriyle yazılır.

Runtime bölümü priority'ye göre event sayısını, en çok tetiklenen Falco kurallarını, gönderilen uyarıları ve açılıp kapanan runtime issue'larını taşır. Ayrıca yürürlükteki runtime policy setini revizyon hash'iyle, dönemdeki policy değişiklikleri, durdurulan pod'lar ve entegrasyon değişiklikleri için `audit_entries`'i, policy ve aksiyona göre eşleşme sayılarını, bir policy'nin işlem yaptığı event'leri (en fazla 500, her biri açtığı issue'yla) ve kapsamayı taşır: entegrasyon açık mı, son event ve event'siz geçen en uzun süre. `event_time` Falco'nun kendi zaman damgasıdır, `received_at` operator'ün event'i aldığı andır. Kapsama `received_at` ile ölçülür, `coverage.clock` bunu belirtir. `runtime.rules[].effect` ve `runtime.decisions[].policy_effect` policy'nin yapmaya ayarlandığı işi verir. Durdurmanın gerçekleşip gerçekleşmediği `runtime.audit_entries` içindeki `runtime_pod_stop` satırındadır.

Harici aktarım ve deny list. 1.4.0'dan itibaren pakette iki bölüm daha vardır. *External export* SIEM hedeflerini, her birinin kabul ettiği son partiyi ve dönemde bırakılan olayları listeler (`report.json` içinde `external_export`). *Deny list* yürürlükteki kayıtları ve dönemde eklenip kaldırılanları, bunların reddettiği ve içeri aldığı admission'ları, istisnaları, her take down ve geri açmayı ve dosya kaydı için dosyayı, hash'ini ve sürümü listeler (`report.json` içinde `deny_list`).



14 Deny list

The deny list holds images, advisories, packages and executables a person decided may not run, each with who added it, when and why; an entry is never deleted, only removed, with who removed it, when and why. It sits in front of the policies: a matching image is refused in every namespace whose policy enforces the list and let in with a warning where it audits it. The entries live in the operator's database, not in the policy, so admission.policy_revisions (the policy hash) does not cover them: the entry that decided a review is named in that admission record's reason ("denied by deny list entry <id>: ...") and listed below. Running workloads are never taken down automatically; every take down is a person's action, listed under takedowns with its reason.

Dönem sonunda yürürlükte 3 kayıt (3 eklendi, 0 kaldırıldı). 75 admission eşleşti: 74 reddedildi, 1 içeri alındı. 1 workload durdurma.

KAYIT	DEĞER	GEREKÇE	EKLENDİ	KALDIRILDI	REDDEDİLEN / ALINAN
#3	digest sha256:083e5fa7fde96dff5efae862e7537dd55eccfec7eecee8ec015b017153e81a99	Worker build a logs card numbers in debug mode. Build d replaces it.	29 Oct 2026, 19:10 UTC · user:usr-c1a1db49a6e0b33a4aff8dc6	—	34 / 0
#2	cve CVE-2024-45491	Heap overflow in libexpat. Rebuild images on expat 2.6.3 or later.	29 Oct 2026, 19:10 UTC · user:usr-c1a1db49a6e0b33a4aff8dc6	—	20 / 1
#1	package bitnami php < 8.2.35	PHP before 8.2.35 is out of support for us. Rebuild legacy images on the 8.2.35 runtime.	29 Oct 2026, 19:05 UTC · user:usr-c1a1db49a6e0b33a4aff8dc6	—	20 / 0

İstisnalar: 1 admissions an entry matched and that were let in because the namespace's policy audits the deny list (denyList: audit); each is also in admission.records

NE ZAMAN	WORKLOAD	NASIL	GEREKÇE	GERİ AÇILDI
29 Oct 2026, 19:14 UTC · user:usr-c1a1db49a6e0b33a4aff8dc6	payments/Deployment/worker	scale_zero	Worker build a writes card numbers to its log. Keep it down until build d is out.	29 Oct 2026, 19:14 UTC

Aynı paketin 14. bölümü, *Deny list*. Dönem sonunda üç kayıt yürürlükteydi. 75 admission'la eşleştiler: 74'ü reddedildi, 1'i namespace'in policy'si listeyi denetlediği için içeri alındı. Her kayıt değerini, gerekçesini, kimin ne zaman eklediğini, reddedilen ve içeri alınan sayılarını gösterir. İstisnaların altında `payments/Deployment/worker` için yapılan take down, gerekçesi ve geri açıldığı zamanla yer alır.

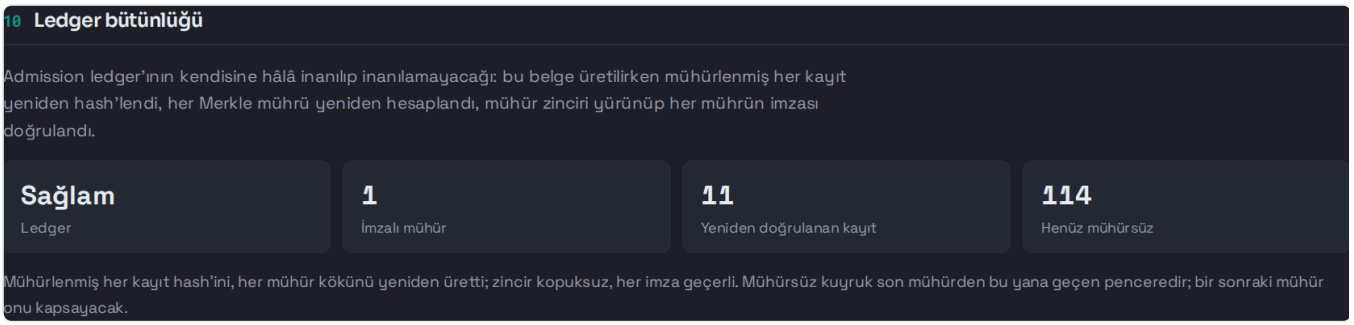
Bulgular digest başına sayılır. 1.4.0'dan itibaren iki referansla, tag'iyle ve digest'iyle ya da iki adla çekilen digest, `findings.critical` içinde ikisini `references` altında yazan tek satırdır ve bulguları `coverage.severity` ile `coverage.findings_total`'da bir kez sayılır. Aynı cluster için şimdi üretilen paket, 1.3.7'nin ürettiğinden tam bu tekrarlar kadar düşük çıkabilir.

Break-glass ve triage. 1.3.4'ten itibaren her break-glass etkinleştirmesi, admission'larının karar verildiği policy revizyonundan okunur. Pencere kapatıldıktan sonra da paket o gün neyin onaylandığını yazar. Geçmiş bir break-glass penceresini kapsayan paketi tam kayıt için yeniden üretin. Triage kararları severity'yi tarama bulgularından alır, bir CVE birden fazla severity ile kaydedildiyse en yükseğini.

Ledger bütünlüğü

Her admission kararı PostgreSQL'e yazılır. Veritabanı kararları sorgulamak için doğru yerdir ama tek başına onlara güvenmek için yeterli değildir. Ledger, değiştirildiğinde bunun anlaşılacağı şekilde kurulmuştur:

- **Her kayıt kendi hash'ini taşır.** İçeriği (zaman, namespace, pod, image, digest, karar, gerekçe, policy adı ve hash'i, break-glass bayrağı) üzerinden yazma anında hesaplanan SHA-256.
- **Kayıtlar imzalı checkpoint'lerle mühürlenir.** Varsayılan olarak saatte bir ya da her 10.000 kayıtta leader replica mühürlenmemiş aralık için Merkle kökü hesaplar ve önceki checkpoint'in hash'ini de içeren bir checkpoint imzalar. Mühürler böylece zincir oluşturur.
- **İmza anahtarı veritabanında değildir.** Cluster'ınızda üretilen ve `attestkeep-evidence-key` Secret'ında tutulan bir Ed25519 anahtarıdır. Ledger satırlarını değiştirmek veritabanına yazma yetkisi, mühürleri yeniden imzalamak bu Secret'a erişim ister. Biri tek başına yetmez. 1.4.0'dan itibaren aynı anahtar kurulumun lisans kimliğidir de (12. bölüm).
- **Kanıt belgelerini de aynı anahtar imzalar**, DSSE zarfı olarak. Lisans sertifikası da anahtarınızı adlandırır, böylece üçüncü bir taraf belgeyi kurulumunuza bağlayabilir.



Paketin içindeki ledger kontrolü. Üretim anında mühürlenmiş her kaydın hash'i yeniden hesaplandı, her Merkle mührü yeniden kuruldu, zincir yürüdü ve her imza kontrol edildi. Burada: *Sağlam*, 1 imzalı mühür ve yeniden doğrulanan 11 kayıt. 114 kayıt son mühürden sonra yazıldı ve henüz mühürlenmemiş sayılır. *Sağlam* bu yüzden mühür sayısıyla birlikte okunur.

Garantinin sınırı

Bu yapı değişikliği görünür kılar, engellemez. En son checkpoint'ten sonra yazılan kayıtlar sayılır ama henüz mühürlü değildir, en fazla bir mühür aralığı kadar. *Sağlam* sonucu mühürlü olan için geçerlidir, checkpoint sayısıyla birlikte okunmalıdır. Hem veritabanına yazma hem imza anahtarına erişimi olan bir saldırgan bu yapıyı aşabilir. İddia, bunun iki ayrı erişim gerektirdiğidir. Denetçi imzalı sonucu ve DSSE zarfını alır. Merkle kontrolünü bağımsız olarak yeniden çalıştırmak veritabanı erişimi gerektirir.

Paketi denetçi için imzalamak JSON'un yanında bir PDF de üretir, ikisi tek bir imzalı arşivde teslim edilir. Teslim edilen paketi herkes attestkeep.com/verify.html adresinde kendi araçlarıyla kontrol edebilir.

11. İşletme, SIEM aktarımı ve yedekleme

Probe'lar

Endpoint	Kontrol ettiği	Kullanımı
`/healthz`	Yalnız process'in ayakta olduğu.	Liveness probe (ilk gecikme 10 sn, periyot 20 sn)
`/readyz`	1.4.0'dan itibaren sunucunun cevap verdiği ve sertifikasının yüklü olduğu. Veritabanı artık bu kontrolde yok: veritabanına ulaşılamazken replica hazır kalır ve webhook statik kurallarla reddetmeye devam eder.	Readiness probe (ilk gecikme 5 sn, periyot 10 sn)

Veritabanı kesintisinde ne olur

- Replica'lar hazır kalır ve cevap vermeye devam eder. Bir admission incelemesi veritabanı işine toplam iki saniye ayırır. Sonra statik kurallar hemen karar verir: `:latest`, registry izin listesi ve digest zorunluluğu reddetmeye devam eder. Kayıt ve cache'e dayanan kararlar zayıflar, kesinti log'a bir kez yazılır.
- Veritabanı `attestkeep_database_up` metriğinde ve sağlık sayfasında görünür. 1.4.0'dan önce veritabanı kesintisini operator hazırlığından izliyorduysanız artık bu metriği izleyin.
- Sonra operator heartbeat boşluğunu bulup kesinti olarak kaydeder, bir sonraki reconciliation taraması kararsız giren her digest'i listeler.
- Bildirimler istek dışında gönderilir, en fazla 64 tanesi bekler. Fazlası bırakılır ve `attestkeep_notifications_dropped_total` ile sayılır.
- Kota sayacı veritabanı hatasında deployment'ı durdurmaz, kabul eder.
- Tarama veritabanıyla durur ve onunla devam eder. Kuyruk veritabanında olduğu için hiçbir şey kaybolmaz.

Alarm verilmesi gerekenler

Metrikler 9090 portunda `/metrics` altında sunulur. Etiketli her seri açılışta sıfırla başlar, bu yüzden verinin yokluğu ile retin yokluğu ayırt edilebilir.

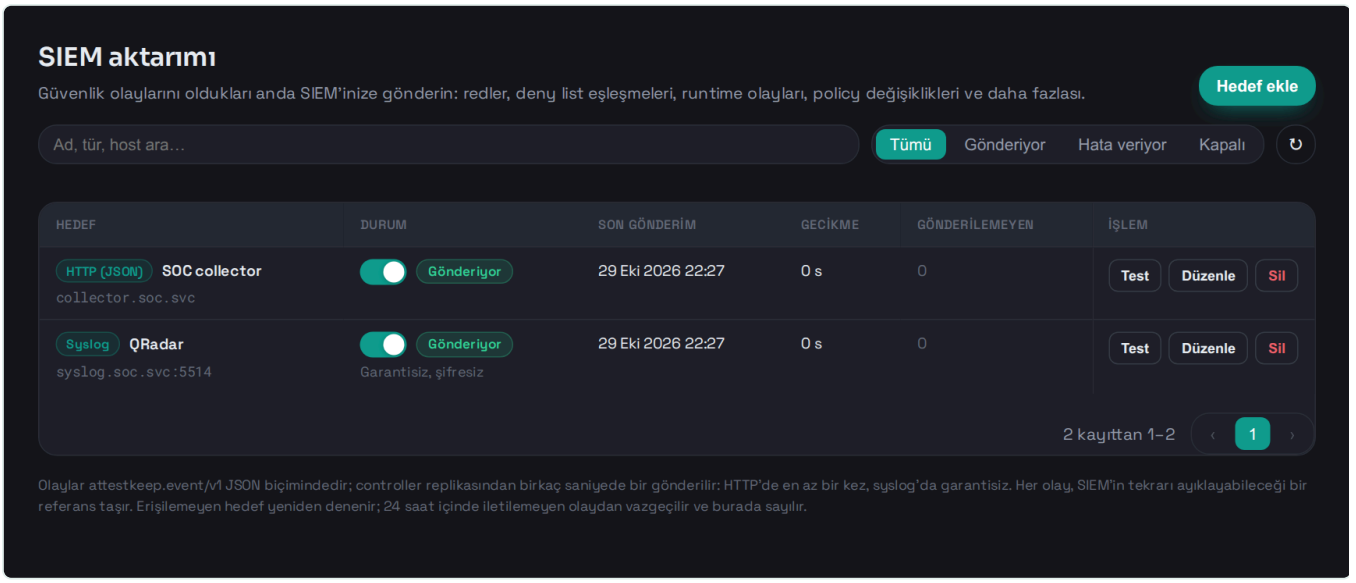
Durum	Sinyal	Neden önemli
Admission atlanıyor	`attestkeep_webhook_distrusted == 1`	`failurePolicy: Ignore` altında bu değer 1 iken deployment'lar kontrolsüz geçer. Alarm olarak ele alın.
İçeri bir şey girdi	`attestkeep_runtime_unmatched_digests > 0`	Reconciliation taraması arkasında izinli admission olmayan çalışan digest'ler buldu.
Veritabanı sorunu	`attestkeep_database_up == 0`	1.4.0'dan itibaren replica'lar kesintide hazır kalır, readiness alarmları bunu artık göstermez.
Tarama geride kalıyor	`attestkeep_scan_queue_depth` sürekli yüksek	Yalnız büyüyen kuyruk, scanner'ın bir registry'ye ya da veritabanına ulaşamadığını gösterir.
Zafiyet verisi eskiyor	`attestkeep_vulnerability_db_age_seconds`	Kararlar arkalarındaki veritabanı kadar günceldir.
Image kotası doluyor	`attestkeep_image_quota{kind="count"}` ve `{kind="limit"}`	Limitte yalnız bu dönem görülmemiş image adları reddedilir, çalışan workload'lar devam eder.
Bir SIEM hedefi geride kalıyor	`attestkeep_siem_lag_seconds{target}`, `attestkeep_siem_events_total{outcome="gave_up"}`	Gecikme, hedefin almadığı en eski olayın yaşıdır. 24 saat sonra hâlâ ulaşmamış olaylar bırakılır.
Node tavanı aşıldı	`attestkeep_nodes{kind="count"}` ve `{kind="ceiling"}`, `attestkeep_node_overage_seconds > 0`	Aşım göstergesi cluster'ın ne kadar süredir tavanın üstünde olduğunu verir. Yeni image adlarının reddedilmeye başlayacağı 14 günden çok önce alarm kurun.

Retler `attestkeep_admission_denials_total{reason}` ile sayılır. Gerekçeler arasında `threshold`, `signature`, `attestation`, `tag`, `digest`, `registry`, `quota`, `unscanned`, `scan_error`, `end_of_life`, `known_exploited`, `provenance` ve `context` var. Node tavanı retleri `quota` altında sayılır. Dashboard için diğer seriler: `attestkeep_vex_statements_total`, `attestkeep_vex_rejected_total`, `attestkeep_issues_opened_total`, `attestkeep_issues_closed_total`, `attestkeep_issues_skipped_total`, `attestkeep_admission_context_total{signal}`, `attestkeep_policy_whatif_runs_total` ve 8. bölümdeki runtime serileri.

Bildirim event'leri de aynı durumları kapsar: image kotası için `threshold_80` ve `threshold_100`, node tavanı için `nodes_80`, `nodes_over`, `nodes_enforced` ve `nodes_cleared`, online sertifikanın bir haftadan az süresi kaldığında `license_expiring`, ayrıca `trivy_db_failed`, `unverified_workload`, `runtime_alert` ve 1.4.0'dan itibaren `deny_list_hit`, `deny_list_entry_added` ve `deny_list_entry_removed`. Yeni bir kanalda `nodes_over`, `nodes_enforced`, Runtime uyarısı ve Deny list eşleşmesi işaretli gelir.

SIEM aktarımı

Güvenlik ekibiniz bu konsola değil SIEM'e bakar. 1.4.0'dan itibaren Ayarlar > SIEM aktarımı, Attestkeep'in güvenlik olaylarını oldukları anda oraya gönderir, böylece topladığınız diğer her şeyle aynı zaman çizgisinde dururlar. Hedef eklemediğiniz sürece hiçbir şey dışarı çıkmaz, hedef eklemek de hiçbir admission kararını değiştirmez. Giriş yapan herkes hedefleri ve her birinin durumunu görür. Hedef eklemek, değiştirmek, test etmek ve kaldırmak yönetici işidir.



İki hedefle Ayarlar > SIEM aktarımı. *SOC collector* HTTP JSON, *QRadar* düz TCP üzerinden LEEF biçiminde syslog alıyor. İkisi de *Gönderiyor*, gecikme yok, vazgeçilen event yok. Syslog satırı düz TCP'nin ne demek olduğunu yazar: garantisiz, şifresiz.

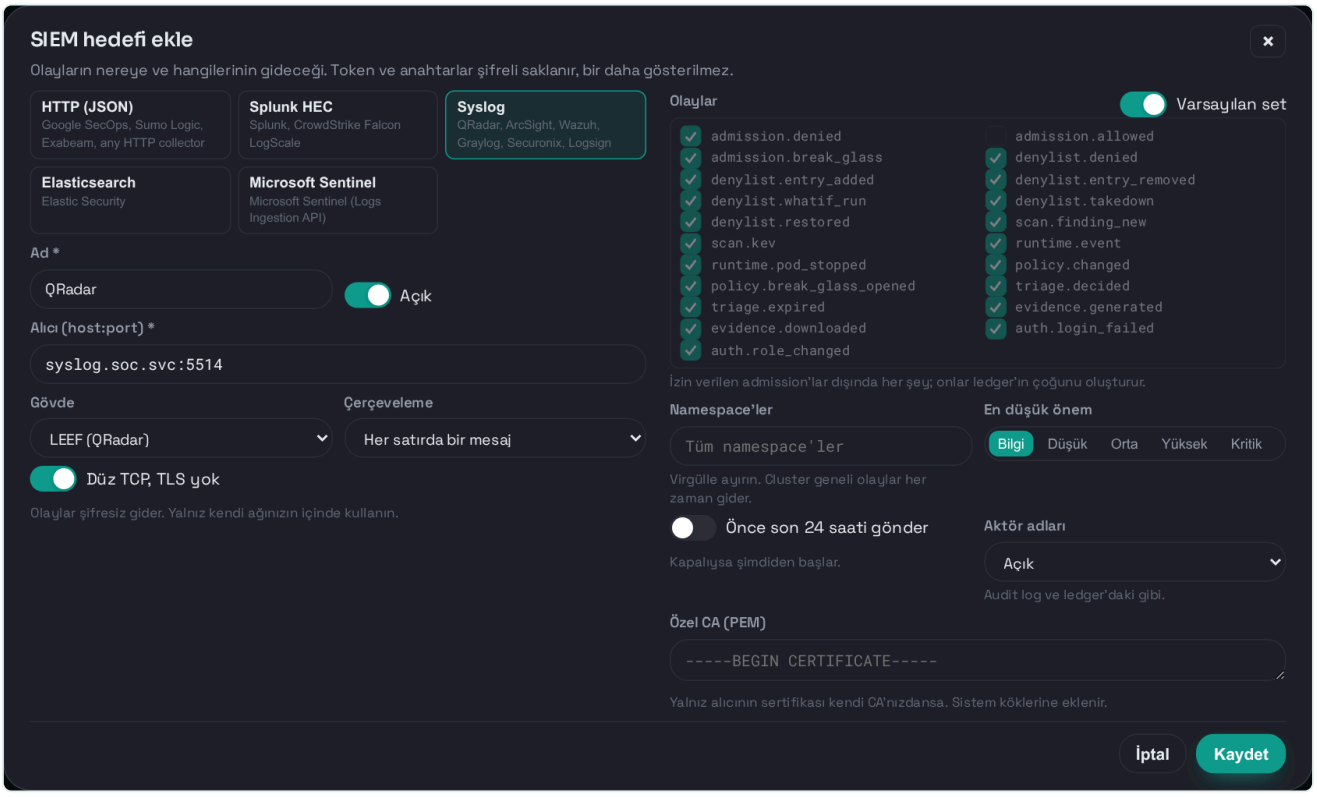
Ne gönderilir. Reddedilen admission'lar, deny list eşleşmeleri, eklenen ve kaldırılan kayıtlar, take down ve geri açmalar, runtime event'leri ve runtime policy'sinin durdurduğu pod'lar, policy, break-glass ve triage değişiklikleri, kanıt üretimi ve indirmeleri, başarısız girişler ve rol değişiklikleri. Kritik bulgu, onu ilk bildiren taramayla bir kez gönderilir. İstismar edilen bulgu da onu KEV listesinde ilk gören taramayla bir kez. İzin verilen admission'lar ledger'ın çoğunu oluşturduğu için siz işaretlemedikçe gönderilmez. Her hedef olay türüne, namespace'e ve en düşük önem derecesine göre süzer. Yeni hedef son 24 saatle başlayabilir.

Olay. Her olay `attestkeep.event/v1` şemasında tek bir JSON nesnesidir: zaman, tür, önem, cluster, namespace, pod, workload, image, digest, policy, aktör, sonuç, gerekçe ve geldiği kayda işaret eden `reference`. Olay yeniden gönderildiğinde reference aynı kalır, SIEM'iniz tekrarları onunla ayıklar. Olay hiçbir zaman SBOM, bulgu listesi, kimlik bilgisi, lisans verisi ya da başarısız girişte yazılanı taşımaz. Hedef, kişileri kanıt paketindeki gibi `user:<hesap id>` ile yazacak şekilde ayarlanabilir.

Hedef	Kimin için	Nasıl
HTTP (JSON)	Google SecOps, Sumo Logic, Exabeam, herhangi bir HTTP collector	Parti başına bir HTTPS POST, JSON dizisi ya da her satırda bir olay. İsteğe bağlı gzip, iki istek header'ı ve `X-Attestkeep-Signature` içinde HMAC imzası.
Splunk HEC	Splunk, CrowdStrike Falcon LogScale	Source `attestkeep` ve sourcetype `attestkeep:event` ile HEC olayları.
Syslog	QRadar, ArcSight, Wazuh, Graylog, Securonix, Logsign	TLS üzerinden RFC 5424, TLS konuşamayan alıcı için düz TCP. Gövde CEF, LEEF ya da JSON.
Elasticsearch	Elastic Security	API key ile bir index'e ya da data stream'e bulk API.
Microsoft Sentinel	Microsoft Sentinel	Bir Entra uygulamasıyla, data collection rule üzerinden Logs Ingestion API.

Teslim. Olaylar HTTP'de en az bir kez, teslim onayı olmayan syslog'da garantisiz gönderilir. Commit sırasıyla, birkaç saniyede bir, istek başına en fazla 100 olay olarak, periyodik işleri yürüten replica'dan gider. Hata veren hedef artan aralıklarla ya da kendi `Retry-After` değerinden sonra yeniden denenir ve diğer hedefleri hiçbir zaman bekletmez. Açık bir veritabanı transaction'ının arkasında bekleyen hedef *Bekliyor* yazar. Olduğu andan 24 saat sonra hâlâ ulaşmamış olay bırakılır ve "N olay gönderilemedi" olarak gösterilir. Kanıt paketinin External export bölümü de bunu listeler. *Test* hedef üzerinden tek bir olay gönderir ve cevabı gösterir.

Kimlik bilgileri. Token, anahtar ve secret'lar kurulumun data key'iyle şifrelenir ve bir daha gösterilmez. Düzenleme hedefin gittiği yeri değiştirirse kayıtlı kimlik bilgileri silinir, böylece siz orada yeniden girmedikçe yeni adrese hiç gönderilmez. Loopback ve link-local adresler, bulut metadata servisi de dahil, reddedilir. HTTP hedefleri `https://` kullanmak zorundadır, sertifikası kendi CA'nızdan gelen alıcı için CA formda verilir.



Syslog seçilmiş *SIEM hedefi ekle*. Alıcı `host:port` olarak yazılır, gövde LEEF (QRadar), her satırda bir mesaj. *Düz TCP, TLS yok* açık, pencere de event'lerin şifresiz gideceğini ve bunun yalnız kendi ağınızın içinde kullanılacağını söylüyor. Varsayılan set izin verilen admission'lar dışında her şeydir. *Önce son 24 saati gönder* kapalı, yani aktarım şimdiden başlar. Aktör adları audit log'da ve ledger'da olduğu gibi açık yazılır. Token ve anahtarlar şifreli saklanır, bir daha gösterilmez.

SIEM aktarımı commit sırasıyla okuduğu için PostgreSQL 13 ya da üstünü ister, chart'la gelen PostgreSQL 16'dır. Daha eski harici sunucuda hiçbir hedef açılamaz ve sayfa *Veritabanı eski* yazar. Hedef yokken de operator'ün kendi log'ları, trace'leri ve metrikleri `observability.otlp.endpoint` ile bir OpenTelemetry collector'a gidebilir. Her SIEM ürünü için kurulum adımları docs.attestkeep.com/siem sayfasındadır.

Neyi yedeklemeli

Attestkeep veritabanı yedeği almaz. Dump'ı almak, zamanlamak ve saklamak sizin işinizdir. Operator'ün bildiği her şey PostgreSQL'de ve cluster kaynaklarındadır. Asıl yedek veritabanıdır: bulgular, kararlar, istisnalar, audit log, kanıt arşivi, kullanıcılar ve lisans sertifikası oradadır.

```
kubectl -n attestkeep exec statefulset/attestkeep-postgresql -- \
  sh -c 'pg_dump -U "$POSTGRES_USER" "$POSTGRES_DB"' > attestkeep.sql
```

Kendi PostgreSQL'inizi kullanıyorsanız mevcut yedekleriniz bunu kapsar. Storage class'ınız destekliyorsa chart'la gelen veritabanının PVC'sinden volume snapshot da alınabilir. Veritabanının yanında üç Secret yedeklenmeli:

- `attestkeep-evidence-key`: kanıt imza anahtarı, operator oluşturur. Kaybedilirse yeni anahtar üretilir, eski anahtarla imzalanmış checkpoint'ler ve belgeler kurulumun yeni anahtarıyla doğrulanmaz. Geçmiş kayıtlar kalır. 1.4.0'dan itibaren lisans kimliği de budur: kaybedilirse bir sonraki günlük kontrol `401` ile cevaplanır ve cluster yeniden aktive edilene kadar Community olarak çalışır.
- `attestkeep-data-key`: `api.dataKey.existingSecret` vermediyseniz chart üretir. Kaybedilirse saklanan tracker token'ları çözülemez, her tracker kanalına token yeniden girilir.

- `attestkeep-session`: oturum anahtarı. Kaybedilirse herkes yeniden giriş yapar.

Webhook sertifikası listede yok, operator onu kendisi yeniden üretir. Aktivasyon anahtarı da yok: aynı cluster'a yeniden kurulum aynı parmak izini verir ve mevcut lisans slot'unu kullanır. Falco entegrasyonu açıkken operator `<release>-falco-mtls` Secret'ını da oluşturur. İçindeki istemci sertifikası Falco > Falco entegrasyonu sayfasından yenilenir.

Ayarlar > Yedekleme, elle kurduğunuz yapılandırmayı da JSON olarak dışa aktarır: bildirim kanalları, triage kararları, kullanıcı listesi, policy'ler ve SIEM hedefleri. İkinci bir cluster'ı aynı şekilde kurmak için kullanılır. Dosyada kimlik bilgisi bulunmaz, içe aktarılan SIEM hedefleri kapalı gelir. İçe aktarım önce önizleme gösterir, mevcut kayıt her zaman korunur, kullanıcılar oluşturulmaz, listelenir. 1.3.1'den itibaren her dışa ve içe aktarım *Backup geçmişi*'nde görünür. Bu dosya veritabanı dump'ının yerini tutmaz.

Geri yükleme kaybolanı gizlemez. Geri yüklemeden sonraki ilk turda operator dump'tan bu yana oluşan heartbeat boşluğunu bulur ve tüm aralığı kontrolün çalışmadığı dönem olarak kaydeder. O dönemin kanıt paketi bu boşluğu diğer kesintiler gibi gösterir. Dump'ı aynı ya da daha yeni bir operator sürümüne geri yüklemek desteklenir, daha yeni bir dump'ı eski operator'e yüklemek desteklenmez.

Erişilebilirlik

Operator ve webhook: 2 replica, tercihli anti-affinity, PodDisruptionBudget minAvailable 1. 1.4.0'dan itibaren kapanan replica boşaltılır: preStop hook'u cluster ona yönlendirmeyi bırakırken `webhook.preStopSleepSeconds` (varsayılan 5) kadar bekler, sonra dinleyicisini beş saniye daha açık tutup elindeki incelemeleri bitirir. Pod'un termination grace period'u bu değer artı 30 saniyedir. Ölçeklemek için `replicaCount`'u artırın. Controller işleri advisory lock ile sıralandığı için bu güvenlidir. Scanner: 1 replica. Chart, ReadWriteOnce cache volume'unda birden fazla replica'yı reddeder ve nedenini söyler. Chart'la gelen PostgreSQL tasarım gereği tek replica'dır. Replikasyon ve yönetilen yedek için `externalDsn` ile kendi PostgreSQL'inizi verin.

GitOps. Enforcement pod admission'ında olduğu için değişikliği kubectl'in, Argo CD'nin, Flux'ın ya da bir Job'un uygulaması fark etmez. GitOps sync'i policy'den geçmeyen bir image'la pod oluşturmaya çalışırsa sync, ret mesajını hata olarak vererek başarısız olur.

İnternete kapalı cluster'lar. Trivy ve cosign image'ın içinde gelir. Zafiyet veritabanını `oras copy ghcr.io/aquasecurity/trivy-db:2 registry.internal/mirror/trivy-db:2` ile kopyalayıp `trivy.dbRepository`'yi ayarlayın. Public transparency log olmadan anahtarla doğrulamak için `scan.cosignOffline`'ı açın. KEV kataloğunun kopyası gzip'lenip `kev.existingConfigMap` ile verilen bir ConfigMap'e konabilir, `kev.source` da `file:///etc/kev/<key>` adresini gösterir. Lisans için ya lic.attestkeep.com:443'e tek bir çıkış yolu ya da offline aktivasyon gerekir (12. bölüm).

12. Lisans

Her satın alma için bir aktivasyon anahtarı, her hesap için bir Community anahtarı verilir. Anahtar planın izin verdiği kadar cluster'a bağlanır, slot'u kendiniz taşıyabilirsiniz.

Aktivasyon

Aktivasyon cluster başına bir kez yapılır ve bir slot kullanır. İlk açılışta operator imza anahtarını cluster içinde üretir, private kısmını bütün replica'ların paylaştığı `attestkeep-evidence-key` Secret'ında tutar, aktivasyon isteğiyle yalnız public kısmını gönderir. 1.4.0'dan itibaren kanıt paketlerini imzalayan da aynı anahtardır, böylece sertifika denetçinin paketleri doğruladığı anahtarı adlandırır. 1.4.0'dan önce aktive edilmiş online kurulum aktive edildiği anahtar çiftiyle devam eder ve bir sonraki aktivasyonunda tek anahtara geçer. İstekte aktivasyon anahtarı, cluster parmak izi (kube-system namespace UID'sinin hash'i, ad ya da adres değil), verdiyseniz cluster adı ve operator sürümü bulunur. Cevap en fazla otuz gün geçerli imzalı bir sertifikadır. Operator onu binary'ye gömülü public key ile cluster içinde doğrular.

1.3.0'dan itibaren operator yalnız node tavanını taşıyan sertifikayı kabul eder. Eski lisans servisinin verdiği sertifika "re-activate with a key issued by licence service 1.3.0 or later" mesajıyla reddedilir.

Operator lisans sunucusuna ulaşamazsa bunu söyler, anahtara geçersiz demez. Güvenmediği bir sertifika, reddedilen ya da zaman aşımına uğrayan bağlantı ve çözülemeyen ad `502` ile, sunucu ve nedenle cevaplanır. Konsol "Lisans sunucusuna ulaşılamadı: sertifikasına güvenilmiyor ya da bağlantı kurulamadı. Sunucu özel bir CA kullanıyorsa chart'ta license.caBundle ayarlayın." yazar. Lisans sunucusundan gelen `5xx` cevabı, "The licence server answered with an error. Try again later." ile `503` olur. Anahtar hiçbir mesajda geçmez. `license.caBundle`, lisans istemcisinin sistem köklerinin yanında güvendiği PEM sertifikalarını alır: TLS'i açan bir proxy ya da özel CA altındaki bir lisans sunucusu için. `--set-file license.caBundle=ca.pem` ile verilir. Sertifika doğrulaması açık kalır.

Günlük kontrol

Kurulum günde bir kez üç değer gönderir: lisans id'si, cluster parmak izi ve operator sürümü, aktivasyondaki anahtarla imzalanmış olarak. 1.4.0'dan itibaren kontrolü tek bir operator replica'sı yapar, scanner artık lisans sunucusuna bağlanmaz. Cevap bir durumdur. Bir şey değiştiyse ya da sertifikanın süresi azaldıysa yeni sertifika da gelir. Image'larınız, workload'larınız, policy'leriniz ya da bulgularınız hakkında hiçbir şey gönderilmez. Telemetri kanalı ve onu açacak bir ayar yoktur. Kaç operator pod'u çalışırsa çalışsın cluster bu çağrıyı günde bir kez yapar.

Lisans servisine ulaşılamazsa

Kurulum elindeki sertifikayla, sertifika bitene kadar çalışır. Başarısız kontrol dört saat sonra yeniden denenir. Online sertifikanın bir haftadan az süresi kaldığında konsol uyarır ve `license_expiring` bildirimi gönderilir. Sertifika biterse kurulum Community Edition olarak devam eder. Durmaz ve deployment reddetmeye başlamaz. `revoked` ya da `released` cevabı bir sonraki kontrolde uygulanır.

Cluster slot'ları

Her aktivasyon, bırakılana kadar bir slot tutar. Slot'ları attestkeep.com'daki lisans sayfanızdan kendiniz bırakırsınız. Bırakılan slot hemen boşalır, onu tutan kurulum bir sonraki günlük kontrolde düşer. Aynı cluster'a yeniden kurulum ikinci slot kullanmaz, parmak izi aynıdır. Slot'lar dolduysa hata, slot'ları tutan cluster'ları ve her birinin aktivasyon tarihini yazar.

Konsoldan da lisans bırakılabilir: Kullanım ve Lisans sayfasında onaydan sonra *Devre dışı bırak*. Online yolda cluster'ın slot'u boşalır. Offline yolda sertifika cluster'dan silinir, slot'u Attestkeep desteği bırakır, çünkü offline cluster geri dönüp haber vermez.

Offline aktivasyon

Lisans servisine hiç ulaşamayacak cluster anahtar yerine sertifikayla aktive edilir. Aktivasyon ekranında ve Kullanım ve Lisans sayfasında bunun için *Sertifika* sekmesi var.



Üç adımda *Sertifika* sekmesi. Bu cluster'ı parmak iziyle, operatörün açık imza anahtarıyla, operatör sürümüyle ve cluster adıyla adlandıran aktivasyon isteğini indirin. İnternete erişen bir makineden attestkeep.com'daki lisans sayfanıza yükleyin. Size dönen sertifikayı yükleyin.

1.4.0'dan önce offline aktive edilmiş kurulum lisansını korur, ama sertifikası eski anahtar çiftini adlandırdığı için imzalı kanıt indirmesi reddedilir. Kullanım ve Lisans'tan alınan yeni aktivasyon isteği yeni bir sertifikayla değiştirilince düzelir. Kanıt sayfası ve ret mesajı bunu söyler.

Sertifika, online yolda saklanan imzalı belgenin aynısıdır, offline olarak işaretlenir. Operator saklamadan önce imzayı, cluster parmak izini ve kendi anahtarını kontrol eder. Bu sertifika için günlük kontrol yapmaz, bitişten otuz gün önce uyarır. Başka cluster'a ya da anahtara verilmiş veya süresi geçmiş sertifika gerekçesiyle reddedilir. Konsolunu kimsenin açmayacağı bir kurulumda `license.existingCertificateSecret`, sertifikayı `certificate` anahtarı altında tutan bir Secret'ı gösterir.

Lisansınız için offline aktivasyonun açık olup olmadığı attestkeep.com'daki lisans sayfasında görünür. Yayımlanan fiyat listesinde yıllık Enterprise ile ya da sözleşmeyle gelir.

Node tavanı

Her plan cluster başına belirli sayıda Ready node'a izin verir. Operator cluster'ın Ready node'larını `nodes.pollInterval` aralığında (varsayılan 5 dakika) sayar. Sayı cluster'da kalır, günlük kontrol yine aynı üç değeri gönderir. Tavanın kendisi sertifikadadır.

Tavanı aşmak hiçbir şeyi durdurmaz. Konsol ve bir bildirim bunu 14 gün sonrasının tarihiyle (`nodes.graceDays`) söyler. Bu 14 gün dolduğunda cluster hâlâ tavanın üstündeyse bu dönem henüz görülmemiş image adları admission'da "node ceiling of N exceeded" mesajıyla reddedilir, reddedilen ad sayılmaz. 1.4.0'dan itibaren image kotasında da böyledir: ayın kotası dolduğu için reddedilen ad sayılmaz, sayaç limitte durur ve *Kota doldu* yazar. Kullanım ve Lisans sayfasındaki lisans kartında dolan image bandı *Süresi dolmuş* değil, lisansın kendi tarihleriyle *Kota doldu* yazar. Kota e-postası da yeni image adlarının admission'da reddedildiğini söyler, artık taranmadıklarını değil. Sayılmış her şey çalışmaya, taranmaya ve denetlenmeye devam eder. Node azaltarak ya da lisansa node ekleyerek tavanın altına inmek sayacı hemen sıfırlar. Node'ları sayamayan bir yoklama hiçbir şeyi değiştirmez. İki chart değeri de tavanı gevşetmez.

Lisans bittiğinde

Abonelik bitip sertifikanın süresi dolduğunda, sertifika doğrulanmaz olduğunda ya da başka bir cluster'a ait çıktığında veya siz devre dışı bıraktığınızda kurulum Community Edition olarak devam eder. Community 1 cluster, 1 framework, ayda 10 image adı ve cluster başına 1 node demektir. Hiçbir şey indirilmez, yeni sertifika üretilmez. Tarama, admission ve kanıt çalışmaya devam eder. Kullanım ve Lisans sayfası ve panel o andan itibaren içinde bulunulan ay için Community haklarını, yani admission'ın uyguladığı sayıları gösterir. Önceki aylar o zamanki haklarıyla kalır. Dönem ortasında iptal, iptal ettiğiniz gün hiçbir şeyi kapatmaz, ödediğiniz dönem sonuna kadar devam eder.

Framework seçimi. Lisansın artık kapsadığından fazla framework için kanıt üretilmişse, giriş yapan ilk owner ya da yönetici lisansın sayısı kadarını seçer. Seçim yapılana kadar kullanılmış framework'lerin hiçbiriyle yeni paket üretilemez, diğer kullanıcılara bir yöneticinin seçmesi gerektiği söylenir. Seçilmeyen framework'lerle yeni paket üretilemez, daha önce üretilen her paket indirilebilir kalır. Seçim lisansın planı ya da framework sayısı değişene kadar geçerlidir ve audit log'a yazılır.

13. Planlar ve fiyatlar

Her güvenlik özelliği ücretsiz planda var. Tarama, imza doğrulama, admission enforcement ve kanıt paketleri Community'de eksiksizdir. DenyUntilScanned ve `failurePolicy: Fail` dahil her policy kontrolü orada çalışır. Deny list, take down, What if ve SIEM aktarımı da öyle. Ücretli planlar kapasite ve single sign-on ekler.

Plan	Aylık, KDV hariç	Yıllık, KDV hariç	Cluster	Framework	Cluster başına image adı / ay	Node / cluster	Ayrıca
Community	$0	$0	1	1, sizin seçiminiz	10	1	Public issue tracker
Team	$600	$6.000	1	1	30	6	Single sign-on, e-posta desteği
Business	$1.800	$18.000	3	3	50	10	Team'deki her şey, öncelikli e-posta desteği
Enterprise	$3.600	$36.000	5	On üçü de	75	20	İnternete kapalı cluster'lar için offline aktivasyon, e-posta desteği

Bir yıl on ay fiyatınadır. Ücretli planlarda aylık eklentiler: ek cluster +$300 (tek checkout'ta en fazla on cluster), ek framework +$120, lisansın kapsadığı her cluster için ayda 10 image adı daha +$30, lisansın kapsadığı her cluster için ek node +$50, birer birer, cluster başına en fazla 50 node. Yıllık fiyatlar aylığın on katıdır. Community'de bir hacim adımı her cluster için ayda +$300'dür. Ek node Community'de satılmaz.

On cluster'dan fazlası, cluster tavanı olmaması ya da node tavanı olmaması liste fiyatıyla değil özel sözleşmeyle olur.

Fiyatlara KDV dahil değildir. Checkout ve ödeme, satıcı sıfatıyla (merchant of record) Polar üzerinden yürür. Kart bilgisi Polar'ın checkout sayfasına girilir ve Attestkeep'in sunucularına ulaşmaz. Vergi varsa fatura adresinize göre orada hesaplanır ve ödemeden önce gösterilir. AB'de KDV mükellefi bir işletme KDV'yi reverse charge ile kendisi beyan eder. Lisans ve destek Attestkeep'ten gelir.

Süreli deneme yok. Community Edition her güvenlik özelliğini taşır ve süresi dolmaz. Ürünü kendi cluster'ınızda istediğiniz kadar deneyebilirsiniz.

Sayılan birim image adıdır: registry ve repository, tag ve digest olmadan. Cluster'da çalışan her şey sayılır, üçüncü taraf image'lar dahil. Kota dolduğunda admission'da yalnız yeni image adları reddedilir. Bu dönem sayılmış her image taranmaya ve denetlenmeye devam eder.

14. Güvenlik, destek ve muhatabınız

Release bütünlüğü

Her release image'ı linux/amd64 ve linux/arm64 için derlenir ve digest'i üzerinden cosign ile imzalanır: GitHub OIDC ile keyless ve docs.attestkeep.com/cosign.pub adresindeki anahtarla. Scanner ve imza doğrulayıcı image'ın içinde gelir, çalışırken indirilmez. Kaynak kod açık değildir.

Hata ya da zafiyet bildirmek

Hatalar, her plan için, github.com/attestkeep/attestkeep-docs adresindeki public tracker'a yazılır. Zafiyetler aynı repository'deki private advisory formuyla ya da security@attestkeep.com adresine bildirilir. Bildirim üç iş günü içinde bir kişi tarafından alındı olarak cevaplanır. Düzeltilen konular, düzeltmeyi taşıyan release çıkınca advisory olarak yayımlanır. Tüzük (EU) 2024/2847 üreticiden zafiyet bildirimleri için tek bir iletişim noktası ister. Bizimki security@attestkeep.com'dur ve aynı bilgi ürünün ürettiği her kanıt paketine basılır.

Plana göre destek

Plan	Destek
Community	Public issue tracker
Team	E-posta
Business	E-posta, öncelikli
Enterprise	E-posta
Özel sözleşme	Sözleşmede yazdığı gibi

Attestkeep hiçbir plan için yanıt süresi garantisi yayımlamaz.

Muhatabınız

Ticari ad	Attestkeep
İşletme adı	Ecavist Technologies, Türkiye'de kayıtlı bir işletme
Kayıtlı adres	Dörtyol Küme Evleri No: 369, İç Kapı No: 3, Işıklar, Merkez, Çanakkale, Türkiye
Hukuki bildirimler, sözleşmeler, fatura	legal@attestkeep.com
Destek ve satış öncesi	support@attestkeep.com
Güvenlik bildirimleri	security@attestkeep.com

Bağlayıcı belgeler (hizmet şartları, mesafeli satış sözleşmesi, gizlilik bildirimi, veri işleme şartları, alt işleyiciler) attestkeep.com'dadır. Sade dille yazılmış bir sayfa ile bir sözleşme çelişirse sözleşme geçerlidir.

Ek. Bu kitabın dayandığı kaynaklar

Ürün sürümü	Attestkeep 1.4.0 (chart 1.4.0, appVersion 1.4.0). Sürüm notlarının tarihi 28 Ekim 2026, image ve chart 29 Ekim 2026'da yayımlandı
Yayımlanan image	`ghcr.io/attestkeep/attestkeep-k8s:1.4.0` `sha256:bfe0299695e38779f02ac24d805088a373d0a754fd7f0f552751e0d9fa3f05e8`
Yayımlanan chart	`oci://ghcr.io/attestkeep/charts/attestkeep` sürüm 1.4.0 `sha256:8d6c4ba1e3ff790f6394e83eb34e5a740eeeecb0b4148e5c7e48683ca9bc610f`
Fotoğraflanan kurulum	Konsolu 1.4.0 sürümünü gösteren, iki node'lu Kubernetes v1.35.1 üzerinde Enterprise Edition olarak çalışan, Falco 0.45.0'ın kendi `http_output` 'u üzerinden mutual TLS ile bağlandığı bir kurulum. Yukarıdaki yayımlanan image'ı değil, 1.4.0 kaynağından derlenmiş bir image'ı çalıştırıyor. 29 Ekim 2026'da Türkçe arayüz ve koyu temayla fotoğraflandı.
Gösterilen kanıt paketi	`PS-SOC2-DEVMINIKUBE-20261001-20261029-DA7379`, 29 Ekim 2026 19:31 UTC'de 1.4.0 sürümüyle üretildi
Dokümantasyon	docs.attestkeep.com'un 1.4.0 sayfaları: releases, install, upgrade, what runs, admission, configuration, deny list, runtime events, SIEM export, licensing, backup, using the console, policies, try a policy before enforcing it, compliance, issue trackers ve operations
Fiyatlar	attestkeep.com/pricing, USD, KDV hariç. Rakamlar 1.3.2 baskısından beri değişmedi, 29 Ekim 2026'da sitenin fiyat listesiyle karşılaştırıldı.

Dokümantasyon: docs.attestkeep.com · Release notları: docs.attestkeep.com/releases · Teslim edilen kanıt paketleri için doğrulayıcı: attestkeep.com/verify.html

1.3.2'den bu yana ne değişti

Sürüm	Değişen
1.4.0, 28 Ekim 2026	Break-glass'tan ve her policy kuralından önce bakılan deny list: her policy'de `denyList` modu, kayıt için What if, workload başına eşleşmeler, etkilenen workload'lar, elle take down ve geri açma (7. bölüm). `attestkeep-binscan` dosya kontrolü image'ların içindeki çalıştırılabilir dosyaları okuma sınırları içinde okur. Beş tür hedefe SIEM aktarımı (11. bölüm). `signatureKeys` ile anahtarlı imza doğrulama. `license.caBundle` , lisans sunucusu hataları artık kendi adıyla. Runtime durdurma, başka ne verilmiş olursa olsun `rbac.allowEnforce` ister. Veritabanı kesintisinde replica'lar hazır kalır, kapanırken boşaltılır. Dry run iz bırakmaz. Kota yüzünden reddedilen image adı sayılmaz, dolan band Kota doldu yazar. Kanıtta 13. ve 14. bölümler, bulgular digest başına bir kez. Lisans ve kanıt için tek anahtar. Go 1.26.10, Trivy 0.76.1, cosign 3.1.4. Migration 0036–0046.
1.3.7, 15 Ekim 2026	`kubectl debug` ile eklenen container'lar admission'da incelenir (3. bölüm). Images sayfası her image'ı dört durumdan birine koyar, yeni Uyumsuz kartı geldi, Engellendi son admission incelemesine bakar. What-if koşuları doksan gün Güvenlik > What-if geçmişi altında saklanır, PDF ve JSON indirmeleri kanıt değil projeksiyondur. Her policy satırında What if, koşunun taranmamış bulduğu image'lar için Şimdi tara, İptal'de kaydedilmemiş değişiklik onayı, cluster'a uygulama yalnız policy yazabilen hesaplara, framework adları tek listeden. Operator Go 1.26.9 ve `golang.org/x/net` 0.60.0 ile derlenir, operator'ün eriştiği yayımlanmış zafiyetler kapanır. Migration 0033–0035.
1.3.6, 8 Ekim 2026	Images sayfası yalnız kabul edilen pod'ları sayar. Falco Event'ler aksiyon filtresi beş etkiyi sunar. Kenar menü ve kanıt listesi 1366 ve 1600 px'te düzeltildi, kullanım göstergesi de. Yeni policy formu dokümante edilen SLA ve triage varsayılanlarıyla açılır.
1.3.5, 8 Ekim 2026	Pod'u durdur olarak ayarlanmış Falco policy'si her yerde böyle etiketlenir, durdurulan pod kanıt paketine `policy:<ad>` olarak yazılır. Cluster chart'taki `clusterName` ile anılır. Denetim Kaydı yerel saat kullanır. Zafiyet veritabanı kartı son kontrol zamanını gösterir ve Şimdi güncelle düğmesi taşır. Webhook kartı gecikmeyi sabit bir sınırla karşılaştırmaz.
1.3.4, 8 Ekim 2026	Kanıt düzeltmeleri: break-glass ayrıntıları o an geçerli policy revizyonundan gelir, triage kararları CVE'nin severity'sini taşır, konsoldaki paket görünümü reddedilen incelemeleri doğru sayar.
1.3.3, 7 Ekim 2026	Migration'lar veritabanı kilidi altında çalışır. Sıfır kurulumda ya da migration getiren yükseltmede bir pod'un bir kez yeniden başlaması artık olmaz.