LUNTRY

luntry.ru

Вебинар

Подпись и валидация образов в Kubernetes

Дмитрий Евдокимов

Founder&CTO Luntry

Станислав Проснеков

Глава DevOps департамента Luntry

CFP

КОНФЕРЕНЦИИ

БЕКОН '25

Принимаем заявки на доклад **до 31 марта 2025**
по темам, связанным с безопасностью контейнеров и Kubernetes

https://bekon.luntry.ru/cfp

whoami

- Основатель и технический директор Luntry
- Опыт в ИБ более 15 лет
- Специализация безопасность контейнеров и Kubernetes
- CFP DevOpsConf, HighLoad++
- Бывший автор статей и редактор рубрик в журнале “ХАКЕР”
- Автор Telegram-канала “k8s (in)security”
- Автор курса “Cloud Native безопасность в Kubernetes”
- Не верит, что систему можно сделать надежной и безопасной, не понимая ее
- Организатор конференции по безопасности контейнеров - БеКон
- Докладчик: BlackHat, HITB, ZeroNights, HackInParis, Confidence, SAS, OFFZONE, PHDays, Kazhackstan, DevOpsConf, DevOops, KuberConf, VK Kubernetes Conference, HighLoad++, БЕКОН и др.

LUNTRY
luntry.ru

whoami

- Глава DevOps департамента Luntry
- Более 10 лет управляет инфраструктурой
- Собирает Luntry с нуля
- Ронял прод

LUNTRY
luntry.ru

О компании Luntry

- Luntry – это Комплексная Защита на всем жизненном цикле контейнерных приложений и средств оркестрации на базе Kubernetes
- Продукт в реестре Минцифры
  - https://reestr.digital.gov.ru/reestr/1057835/
- В процессе получения сертификата ФСТЭК
  - 1 квартал 2025

LUNTRY
luntry.ru

Функциональность Luntry

Валидация образов

- Контроль источника образов
- Контроль подписи образа

"Вебинар Luntry. С чего начать защиту кластера Kubernetes?"

План вебинара

- Валидация образов контейнеров
  - Почему? Зачем? Как?
  - Подготовка инфраструктуры
- Взгляд Luntry

Валидация образов контейнеров

Cloud Native Security Whitepaper v2

practices, and other malfeasance. Upon completing these checks, it is important to cryptographically sign artifacts to ensure integrity and enforce non-repudiation. Immutable image binary and immutable URL of image is also worth noting

for the real-time and continuous validation of workload attributes e.g. signed artifacts are verified, container image and runtime security policies are ensured,

Supply chain tools can gather and sign build pipeline metadata. Later stages can then verify the signatures to validate that the prerequisite pipeline stages have run.

Artifact Registries

Registries should accommodate technologies to sign and verify OCI artifacts. It is also important to ensure that the caching and distribution tools also provide the capability to sign, encrypt and provide checksums to ensure that the caching layer can detect tampering or attempts to poison the dataset.

Signing, Trust, and Integrity

Digital signing of image content at build time and validation of the signed data before use protects that image data from tampering between build and runtime, thus ensuring the integrity and provenance of an artifact. Confirmation starts with a process to indicate that an artifact was vetted and approved. The trust confirmation also includes verifying that the artifact has a valid signature. In the simplest case, each artifact can be signed by one signer to indicate a single testing and validation process that the artifact has gone through. However, the software supply chain is more complex in most cases, and creating a single artifact relies on multiple validation steps, thus, depending on a conglomerate of entities' trust. Examples of this are:

- Container image signing - the process of signing a container image manifest
- Configuration file signing - signing of a config file, i.e. application config files
- Package signing - Signing of a package of artifacts, like application packages

For generic software artifacts such as libraries or OCI artifacts, signing these artifacts indicates their provenance of approved usage by the organization. Verification of these artifacts is equally crucial in ensuring that only the authorized artifacts are allowed. It is strongly recommended that repositories require mutual authentication to introduce changes to images in registries or to commit code to repositories.

Pre-Flight Deployment Checks

Before deploying a container image, organizations should verify the existence, applicability, and current state of:

- Image signature and integrity
- Image runtime policies (e.g absence of malware or critical vulnerabilities)
- Container runtime policies (e.g absence of excessive privileges)
- Host vulnerability and compliance controls
- Workload, application, and network security policies

Image Trust & Content Protection

Utilization of a policy agent to enforce or control authorized, signed container images allows organizations to provide assurance of the image provenance for operational workloads. Further, inclusion of encrypted containers allows for the

dence should be cryptographically verified when possible. The software producer should use trusted documents such as signed meta-data documents and signed payloads to verify the authenticity and integrity of the built environment.

The CI/CD system should sign both application and container images with signatures reproduced in the SBOM. Operators may use post-build SBOM gener-

CIS Kubernetes Benchmark

5.5.1 Configure Image Provenance using ImagePolicyWebhook admission controller (Manual)

Profile Applicability:

- Level 2 - Master Node

Description:

Configure Image Provenance for your deployment.

Rationale:

Kubernetes supports plugging in provenance rules to accept or reject the images in your deployments. You could configure such rules to ensure that only approved images are deployed in the cluster.

Impact:

You need to regularly maintain your provenance configuration based on container image updates.

Audit:

Review the pod definitions in your cluster and verify that image provenance is configured as appropriate.

Remediation:

Follow the Kubernetes documentation and setup image provenance.

Default Value:

By default, image provenance is not set.

"Вебинар Luntry. Соответствует ли ваш Kubernetes-кластер лучшим практикам?"

Модели нарушителя

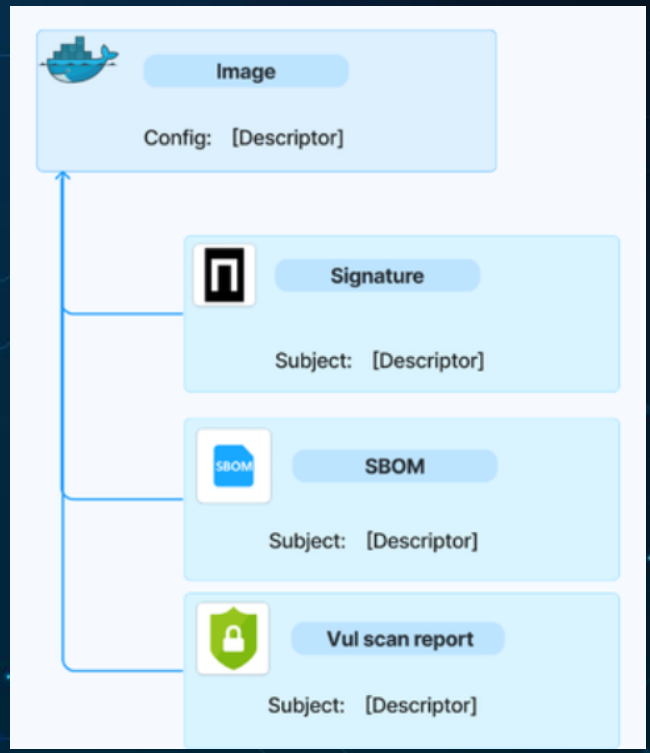
- Внешний нарушитель
  - Донести образ контейнера с необходимым инструментарием
- Внутренний нарушитель
  - Донести образ контейнера с необходимым инструментарием
- Скомпрометированный или вредоносный разработчик
  - Попытка обойти проверки безопасности, security/quality gate и т.д.

OCI спецификации

- Open Container Initiative Image Format
  - https://github.com/opencontainers/image-spec
  - Текущая последняя версия 1.1.0
- Open Container Initiative Distribution Specification
  - https://github.com/opencontainers/distribution-spec
  - Текущая последняя версия 1.1.0
- Open Container Initiative Runtime Specification
  - https://github.com/opencontainers/runtime-spec
  - Текущая последняя версия 1.2.0

Безопасность цепочки поставки в 1.1

- Связывание метаданных цепочки поставок с образами контейнеров
  - Для повышения прозрачности, целостности и безопасности пользователи могут связать метаданные цепочки поставок и аттестации с образом контейнера
  - OCI-Conformant Products



Типы проверок

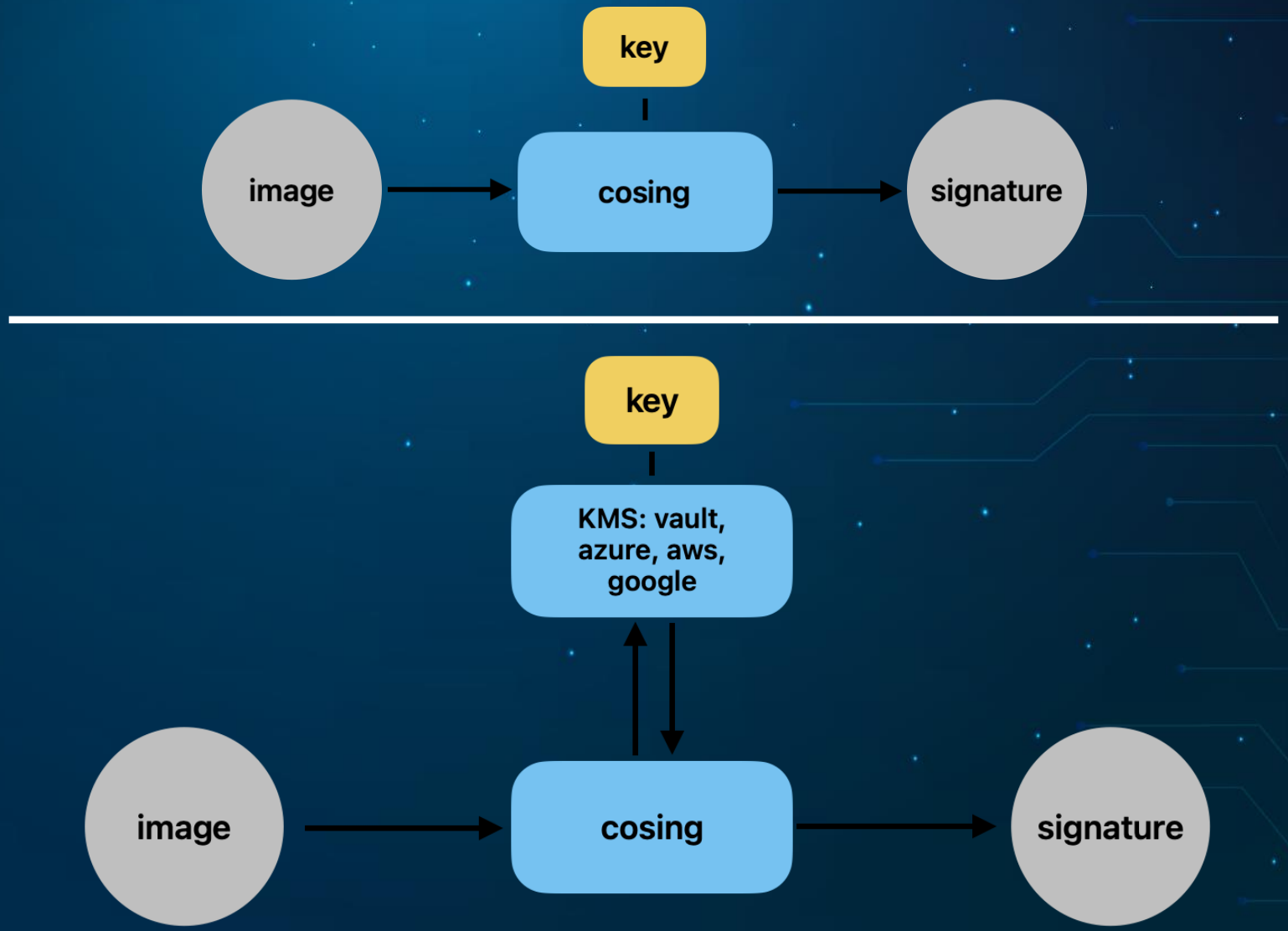
- По image registry
  - Проверка на разрешенный список registry
    - Отдельный или общий список registry на Namespace
      - На пример, в коммунальном кластере в разные Namespace можно запускать сервисы только из определенных registry
- По подписи
  - Проверка на валидность подписи
    - Отдельная или общая подпись на Namespace
      - На пример, отдельные подписи на разные окружения (dev, test, stage, prod)

Pipeline в общем виде (Где push ?!)

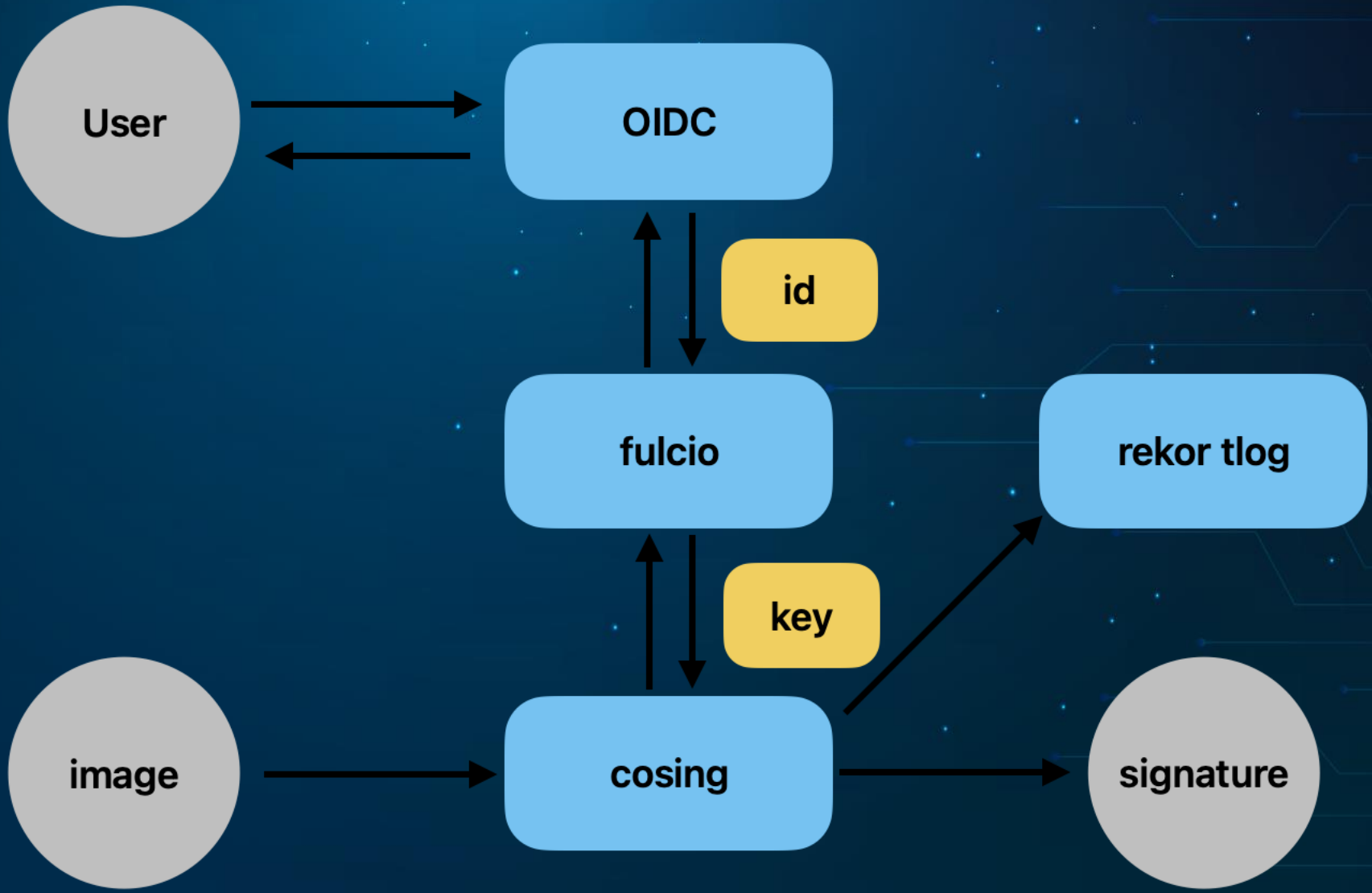
Проблема TOCTOU

- Time-of-check и Time-of-use
  - Изменяемость tags образов
  - Образ, проверяемый во время конвейера CI/CD или фазы деплоя в Kubernetes, отличается от образа, развернутого в кластере
  - Необходимо использовать digest вместо tags

Хранение ключей



Хранение ключей



Безопасность цепочки поставки с Sigstore

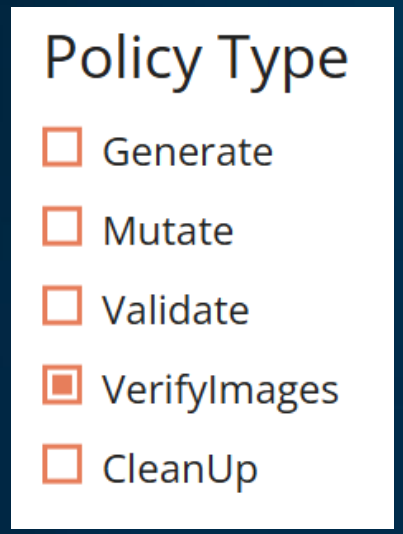
- cosign - Code signing and transparency for containers and binaries
- fulcio - Sigstore OIDC PKI
- rekor - Software Supply Chain Transparency Log

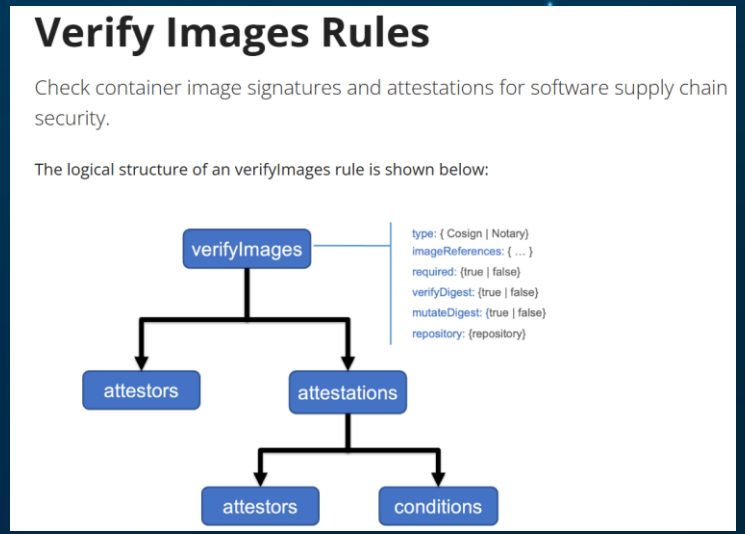
Специализированные инструменты

- In-toto – фреймворк для защиты целостности цепочки поставки
- Notary – проект для подписи и верификации данных
- Portieris – Admission Controller для верификации образов
- Connaisseur – Admission Controller для верификации образов
- policy-controller – Admission Controller для верификации образов

Policy Engines

- OPA Gatekeeper
  - cosign-gatekeeper-provider – интеграция с cosign через фичу External Data
- Kyverno
  - https://kyverno.io/policies/?policytypes=verifyImages






```yaml
rules:
  - name: verify-image
    match:
      any:
      - resources:
          kinds:
            - Pod
    verifyImages:
    - imageReferences:
      - "ghcr.io/kyverno/test-verify-image*"
      mutateDigest: true
      attestors:
      - entries:
        - keys:
            publicKeys: |
              -----BEGIN PUBLIC KEY-----
              MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE8nXRh950IZbRj8Ra/N9sbqOPZrfM
              5/KAQN0/KjHcorm/J5yctVd7iEcnessRQjU917hmKO6JWVGHpDguIyakZA==
              -----END PUBLIC KEY-----
```

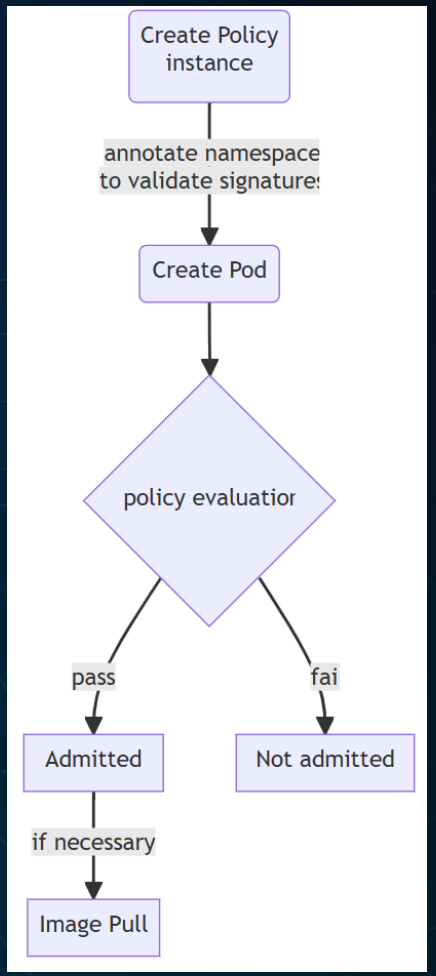
Проверка подписи на CRI Runtime

Подписи образов контейнеров можно проверять прямо на Node возможностями Container Runtime.

1) CRI-O с версии 1.28 может самостоятельно проверять подписи образов.

2) В Kubernetes версии 1.28 появилась новая image pull ошибка с кодом SignatureValidationFailed

Про аналогичную возможность на стороне containerd пока ничего неизвестно.



Проверка подписи артефактов Kubernetes

- Раздел документации “Verify Signed Kubernetes Artifacts”

Verifying images for all control plane components

To verify all signed control plane images for the latest stable version (v1.31.0), please run the following commands:

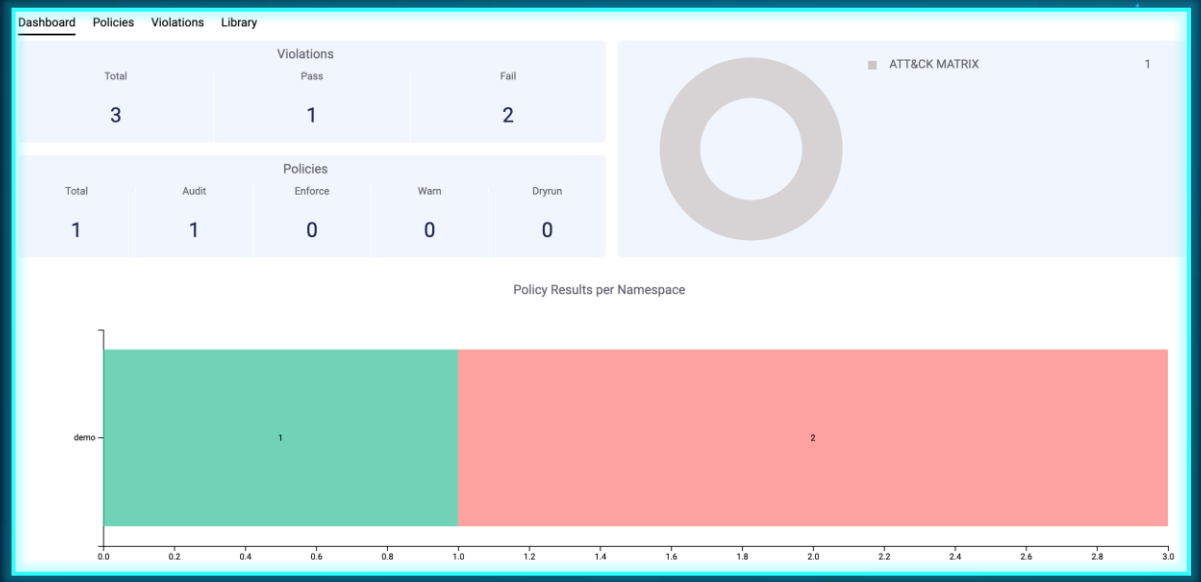
```
curl -Ls "https://sbom.k8s.io/$(curl -Ls https://dl.k8s.io/release/stable.txt)/release" \
  | grep "SPDXID: SPDXRef-Package-registry.k8s.io" \
  | grep -v sha256 | cut -d- -f3- | sed 's/-/\//' | sed 's/-v1/:v1/' \
  | sort > images.txt
input=images.txt
while IFS= read -r image
do
  cosign verify "$image" \
    --certificate-identity krel-trust@k8s-releng-prod.iam.gserviceaccount.com \
    --certificate-oidc-issuer https://accounts.google.com \
    | jq .
done < "$input"
```

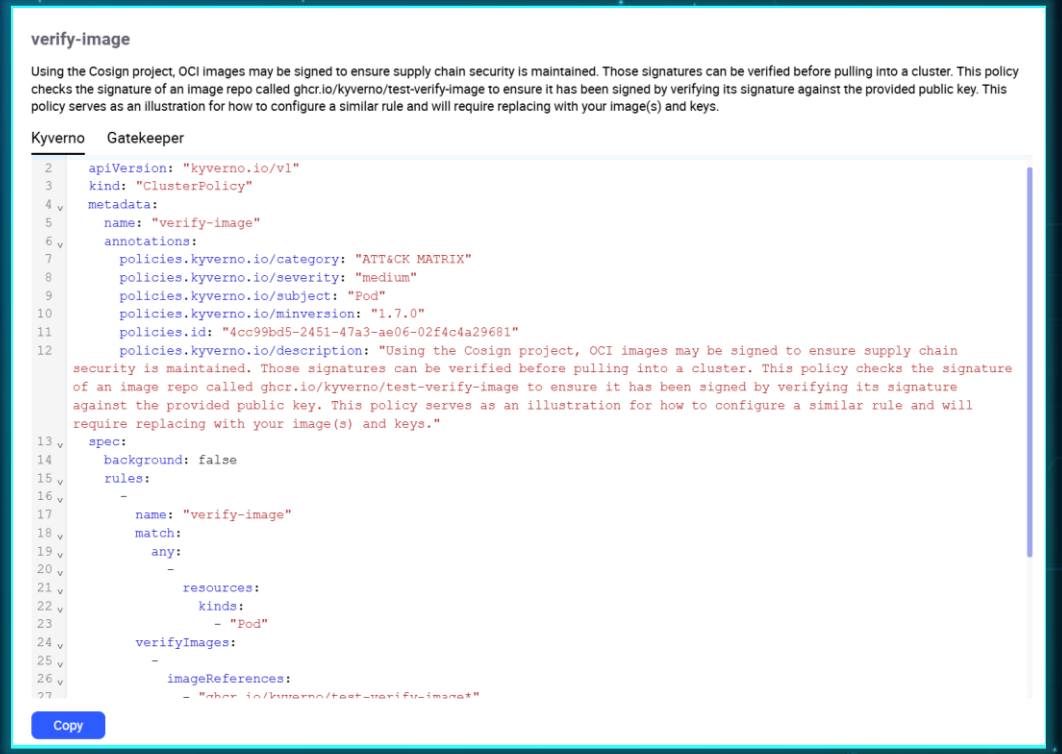
Взгляд Luntry

Подход Luntry

- Переиспользуемость
  - Интеграция с Policy Engine: Kyverno и OPA Gatekeeper
- Прозрачность
  - Использование подход Policy-as-Code
- Гибкость
  - Проверка микросервисов в режиме background scan
- Надежность
  - Надежность уровня Policy Engine
- Функциональность
  - Проверка по image registry и по подписи с учетом namespaces
- Максимальная нативность
  - Использование стандарта де-факто cosign в CI\CD

Скриншоты Luntry





Итог

1. Контроль происхождения образов чрезвычайно важен
2. В подписи и валидации образов в K8s нет ничего сложного
3. Luntry сильно упрощает контроль данного аспекта безопасности

Спасибо за внимание!

Дмитрий Евдокимов
Founder&CTO

Email: de@luntry.ru
Twitter: @evdokimovds
@Qu3b3c
Channel: @k8security
Site: www.luntry.ru

Станислав Проснеков
Глава DevOps

Email: sp@luntry.ru

k8security luntrysolution