



O T U S

Онлайн образование

otus.ru

 Проверить, идет ли запись

Меня хорошо видно && слышно?



Основные модели безопасности и контроллеры в Kubernetes



Игнатенко Филипп

Преподаватель на курсе “DevOps практики и инструменты”

Преподаватель



Игнатенко Филипп

- Руководитель блока развития российской облачной платформы
- Преподаватель курса “DevOps практики и инструменты” в OTUS, а также курсов по Linux, Docker, DevSecOps, Kubernetes
- Автор образовательных программ для министерства образования
- Спикер международных конференций



Маршрут вебинара

1. Безопасность docker

2. Admission controllers

3. Безопасность хоста

4. Безопасность в runtime

5. Network security

6. Application security

7. DEMO
(network policy + cilium-editor)

9. DEMO
(kube-hunter)



Правила вебинара



Активно участвуем



Задаем вопросы в чат



Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения



Пишем в чат



Документ



Ответьте себе или
задайте вопрос

Цели вебинара

После занятия вы сможете

1. Узнать об основных векторах защиты k8s-кластера

2. Подробнее остановиться на каждом направлении по защите кластера

3. Закрепить на практике:
 - работу с сетевыми политиками в кластере
 - использованию редактора cilium-editor
 - проведение внешнего и внутреннего сканирования кластера с помощью kube-hunter

3. Сформировать комплексное видение обеспечения информационной безопасности кластера kubernetes

Основы безопасности k8s

Безопасность k8s осуществляется на следующих основных этапах:

- Безопасность docker image
- Admission controllers
- Безопасность хоста (node)
- Безопасность в рантайме
- Network security
- Application security



Безопасность docker

Безопасность docker

Базовые правила по обеспечению безопасности docker:

- Регулярно устанавливайте обновления
- Ограничивайте доступ к сокету docker и не оставляйте его без защиты
- Создавайте и используйте непривилегированного пользователя для работы контейнера
- Не запускайте контейнер с повышенными привилегиями
- Используйте профильные инструменты для проверки docker-образов на предмет уязвимостей
- Ограничивайте возможности контейнера и его ресурсы
- Предотвращайте повышение привилегий внутри контейнера
- Отключайте межконтейнерное взаимодействие
- Используйте модули безопасности Linux
- Используйте readonly ФС и docker volumes



Безопасность docker

Дополнительные правила по обеспечению безопасности docker:

- Используйте легковесные образы ОС
- Подписывайте и проверяйте docker-образы для митигации атак MITM
- Используйте доверенные хранилища артефактов
- Применяйте механизм многоэтапной сборки (multistaging)
- Не допускайте утечки сенситивной информации в Dockerfile
- Используйте фиксированные теги версий образов
- Используйте инструкцию COPY вместо ADD
- Используйте линтеры при работе с Dockerfile
- Используйте полноценный DevSecOps-конвейер



Admission controllers

Admission controllers

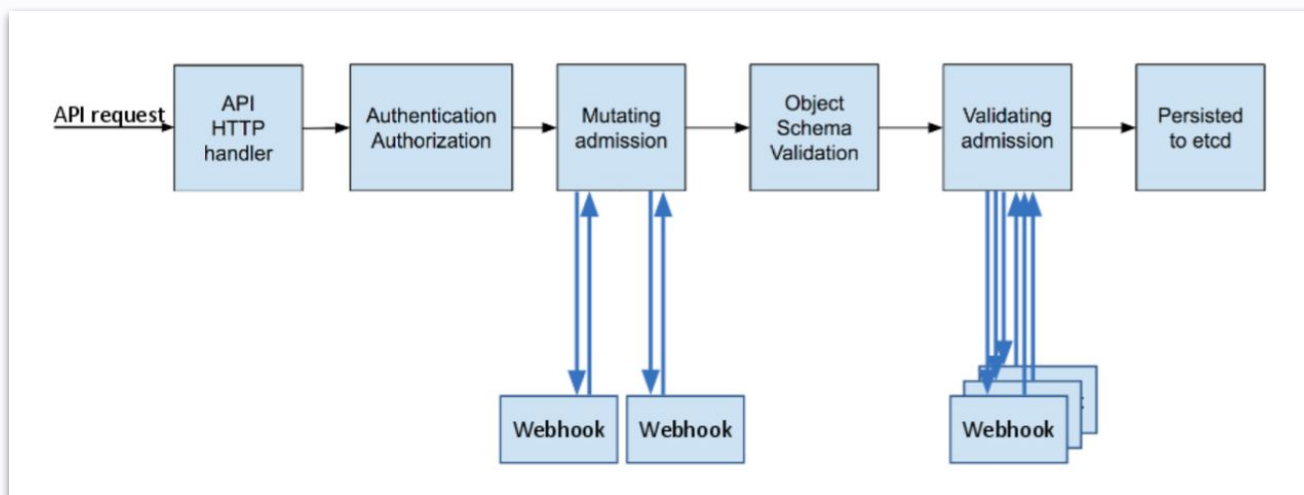
Admission controllers - это специальные **плагины** k8s, которые перехватывают запросы к серверу API Kubernetes **до сохранения объекта**, производя проверку подлинности и авторизации запроса



Admission controllers

Концепция admission controllers состоит из следующих **этапов**:

- API HTTP handler
- Authentication/Authorization
- Mutating admission (mutating webhooks)
- Schema validation
- Validating admission (validating webhooks)
- Persisted to etcd



Admission controllers

Включение admission controller'ов может происходить посредством отправки **флага** напрямую kubeapi-server:

```
--enable-admission-plugins=ValidatingAdmissionWebhook,MutatingAdmissionWebhook
```

Kubernetes советует использовать следующие admission controller'ы **по умолчанию**:

```
--enable-admission-plugins=NamespaceLifecycle,LimitRanger,ServiceAccount,DefaultStorageClass,DefaultTolerationSeconds,MutatingAdmissionWebhook,ValidatingAdmissionWebhook,Priority,ResourceQuota,PodSecurityPolicy
```

Admission controllers

С помощью валидирующих и модифицирующих вебхуков, можно внедрить **запрет запуска пода** под повышенными **привилегиями**.

При попытке развернуть под с повышенными привилегиями мы **получим ошибку**, и такой под **не будет** развернут:

```
$ kubectl create -f examples/pod-with-conflict.yaml
Error from server (InternalError): error when creating "examples/pod-with-conflict.yaml":
Internal error occurred: admission webhook "webhook-server.webhook-demo.svc" denied the request:
runAsNonRoot specified, but runAsUser set to 0 (the root user)
```


Какие преимущества дает использование и модификация admission controller'ов?

Безопасность. Admission controllers могут повысить безопасность, установив разумный базовый уровень безопасности для неймспейса или всего кластера.

Встроенный **PodSecurityPolicy** admission controller можно использовать, например:

- Для **запрета запуска** контейнеров **под root'ом** или обеспечения того, чтобы корневая файловая система контейнера всегда монтировалась только для чтения
- Разрешить извлечение образов **только из определенных** реестров, известных предприятию
- **Отклонять** развертывания, **не соответствующие** стандартам безопасности. Например, контейнеры, использующие privileged-флаг, могут обойти множество проверок безопасности

Какие преимущества дает использование и модификация admission controller'ов?

Управление конфигурацией. Admission controllers позволяют проверять конфигурацию объектов, работающих в кластере, и предотвращают попадание каких-либо очевидных неправильных конфигураций в ваш кластер.

Admission controllers могут также быть полезны для обнаружения и исправления образов, развернутых без семантических тегов, например:

- автоматическое **добавление лимитов** ресурсов или проверка лимитов ресурсов
- автоматическое **добавление меток** и аннотаций
- проверка того, чтобы ссылки на образы, используемые в продакшен развертываниях, **не использовали latest-теги** или теги с -dev-суффиксом

Безопасность хоста (node)

Базовые принципы безопасности хоста

- Регулярно устанавливайте **обновления** (ОС, библиотеки, зависимости)
- Используйте инфраструктурные **сканеры** (nessus, openvas greenbone, xspider)
- По возможности, не используйте public-ip (используйте внутренние сети или VPN)



Безопасность в рантайме

Pod Security Policy (PSP)

Pod security policy (deprecated in v1.21) являлись возможностью admission controller'ов, и позволяли добиваться следующих преимуществ в обеспечении информационной безопасности кластера:

- **запрет** запуска подов **под рутом**
- **невозможность** записи в корневую файловую систему
- **запрет** на **монтирование** каталога с хоста системы
- **запрет** на порождение **новых процессов** и тд

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: example
spec:
  privileged: false # Don't allow privileged pods!
  # The rest fills in some required fields.
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  volumes:
    - *
```

Pod Security Admission (PSA)

На смену устаревающему **Pod Security Policy** (**deprecated in v1.21**) пришли **Pod Security Admission (PSA)**

PSA можно активировать начиная с **v1.22**:

```
--feature-gates="..., PodSecurity=true"
```

Начиная с версии **v1.23** плагин включен **по умолчанию**

Pod Security Admission (PSA)

После активации **PSA** (или установки веб-перехватчика) вы можете настроить пространства имен, чтобы определить режим контроля доступа, который вы хотите использовать.

Kubernetes определяет набор режимов, которые вы можете установить, чтобы определить, какой из предопределенных уровней безопасности Pod Standard вы хотите использовать для неймспейсов.

Выбранная вами метка определяет, **какое действие** предпринимает плоскость управления при обнаружении потенциального нарушения

Sysdig Falco

Sysdig Falco — opensource-инструмент для обнаружения **аномалий** и **мониторинга активности** в системе. Работает как на хосте, так и в контейнерах, если потребуется.

Falco состоит из **двух компонентов** — **модуль ядра** `falco_probe`, и непосредственно сам **демон**, который обрабатывает собранную информацию, генерирует отчёты и т. д. Настройка выполняется из нескольких `.yaml` файлов



Sysdig Falco

Falco позволяет задействовать список **разрешенных действий**, например:

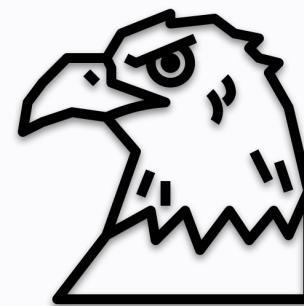
- какие процессы и команды могут быть запущены
- какие процессы могут устанавливать соединения
- какие порты могут прослушиваться
- в какие файлы и каталоги можно писать
- какие вызовы ядра могут выполняться

Что это поможет **предотвратить**?

- запуск shell в контейнере пода
- монтирование каталога с хост машины
- возникновение нового процесса
- чтение файлов
- установление непредвиденного сетевого соединения

что с этим **можно сделать**:

- syslog
- shell (вызов телеграм, слак, почты)

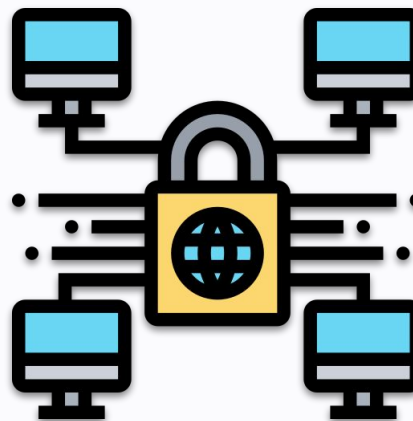


Network security

Network security

В кластере k8s по умолчанию наблюдаются **следующие проблемы** с сетевой безопасностью:

- трафик в кластере по умолчанию **разрешен** и никак **не контролируется** (если не используется, к примеру, service mesh)
- адреса подов часто **меняются**
- трафик **не шифруется** (если не используются дополнительных средств, service mesh в том числе)



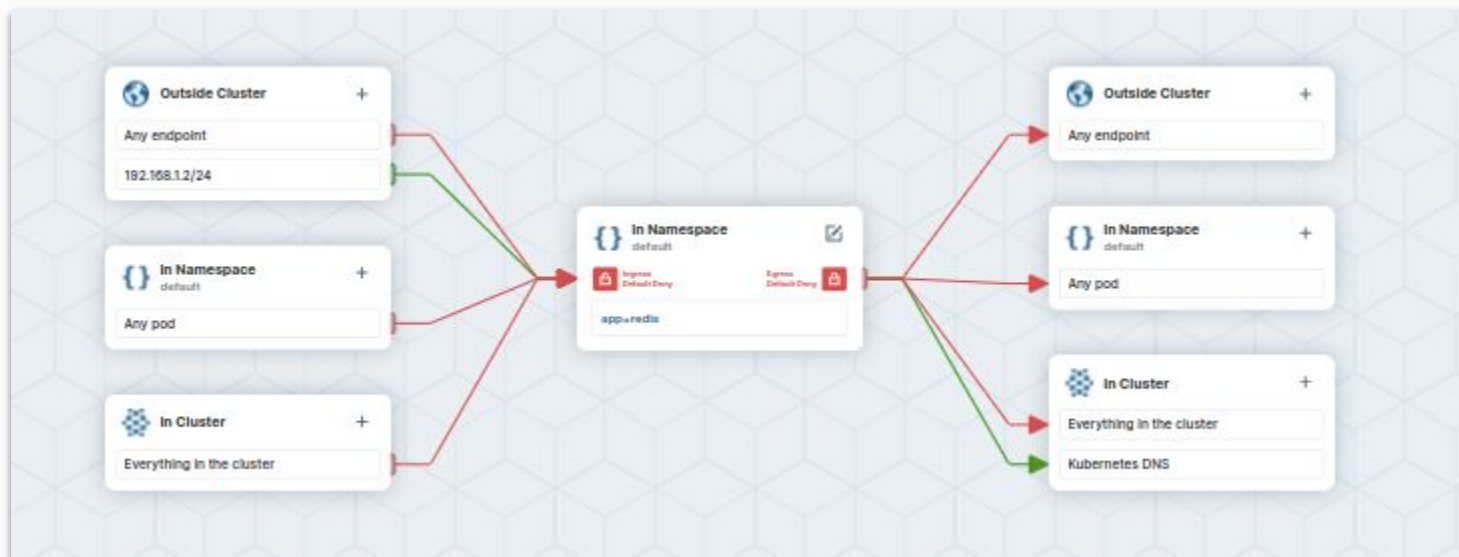
Network security

Для **ограничения доступа** к рабочим нагрузкам (например, к службе nginx), чтобы только помеченные поды могли запрашивать её, создайте объект **NetworkPolicy**:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: access-nginx
spec:
  podSelector:
    matchLabels:
      app: nginx
  ingress:
    - from:
      - podSelector:
          matchLabels:
            access: "true"
```

Network security

Для удобства создания манифестов для желаемых **Network Policy** можно использовать инструмент **cilium editor** (cni cilium) - <https://editor.cilium.io/>















Network security

Network Policy позволяют настроить **ingress** и **egress** правила для ресурсов неймспейса на основе правил (меток)

Функциональность **network policy** (с поддержкой шифрования) предоставляется сетевыми плагинами (**cni** — container network interface)

Но **не все cni-плагины** поддерживают network policy и шифрование!

<i>CNI</i>	<i>ENCRYPTION</i>	<i>NETWORK POLICIES</i>
Calico	 No	 Ingress + Egress
Canal	 No	 Ingress + Egress
Cilium	 Yes	 Ingress + Egress
Flannel	 No	 No
Kube-router	 No	 Ingress only
WeaveNet	 Yes	 Ingress + Egress

Service mesh

В обеспечении безопасности сетей в кластере вам также может помочь сервисная сетка (**service mesh**)

Сервисная сетка – это набор инструментов, позволяющих осуществлять мониторинг трафика между микросервисами, включая схему взаимодействия и коды HTTP-статусов между ними.

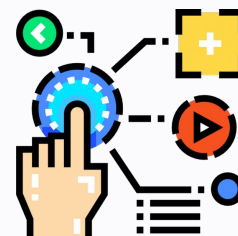
Позволяет использовать **mTLS** для шифрования HTTP-трафика между сервисами



Service mesh

Преимущества сервисной сетки (**service mesh**) на базе инструмента **Istio**:

- Istio – вмешательство в сетевое взаимодействие
- Отказоустойчивость: ретрай запроса при сбое (благодаря коду ответа)
- Канареечные выкаты: регулирует процент сетевого трафика на новой версии сервиса
- Управление трафиком (таймауты, ретраи, балансировка)
- Наблюдаемость: журналы, метрики, трассировки
- Безопасность: аутентификация и авторизация, работа с токенами, контроль трафика



Application Security

App security

Для обеспечения безопасности развернутых в вашем кластере k8s приложений воспользуйтесь следующими **рекомендациями**:

- Используйте **k8s secrets** (или HashiCorp **Vault** в качестве альтернативы)
- Применяйте практики **devsecops** (**SAST, SCA, DAST, IAST, RASP**)
- Используйте сервисную сетку (**service mesh**) для контроля за интеграционными потоками приложений, расширенными возможностями деплоя и обновления)
- Проводите аудит кластера с помощью профильных инструментов проверки: **kube-hunter**, kube-audit, chaos-monkey и проч.



DEMO

(network policy + cilium-editor)

DEMO (kube-hunter)

Рефлексия

Цели вебинара

Проверка достижения целей

1. Узнали об основных векторах защиты k8s-кластера

2. Подробнее остановились на каждом направлении по защите кластера

3. Закрепить на практике:
 - работу с сетевыми политиками в кластере
 - использованию редактора cilium-editor
 - проведение внешнего и внутреннего сканирования кластера с помощью kube-hunter

4. Сформировали комплексное видение обеспечения информационной безопасности кластера kubernetes

Следующий вебинар

Тема: «Ingress-контроллеры и сервисы в Kubernetes»



04.04.23 (Вторник, 04 апреля)



Ссылка на вебинар будет в ЛК за 15 минут



Материалы к занятию в ЛК — можно изучать



Обязательный материал обозначен красной лентой

Заполните, пожалуйста,
опрос о занятии

Спасибо за внимание!