VK Cloud

Защита контейнеров и микросервисной разработки

Станислав Погоржельский,
технологический евангелист
VK Cloud

tech

Проблематика →

Уязвимости в контейнерах и атаки на кластеры Kubernetes становятся всё более частыми, подвергая риску инфраструктуру и данные. В таких условиях безопасность контейнерной среды — это не опция, а необходимость для сохранения контроля, быстрой реакции и восстановления после инцидентов

tech

О чем говорим?

01 Как меняются стандарты безопасности в Kubernetes и с какими новыми вызовами сталкиваются пользователи

02 Как построить процесс защиты контейнеров Kubernetes: от образов контейнеров до кластера и сети

03 Как настраивать контроль доступа для пользователей и отслеживать подозрительные активности

Новые стандарты безопасности в k8s

tech

Критические уязвимости (CVE) за 2024–2025 годы

CVE-2024-21626:
Уязвимость в runC, позволяющая контейнерам получить доступ к файловой системе хоста, что может привести к побегу из контейнера

CVE-2024-3094:
В XZ Utils, внедренная в процессе сборки, может быть использована для удалённого выполнения кода

CVE-2024-31989:
Уязвимость в Argo CD, связанная с отсутствием пароля в Redis по умолчанию, что может привести к эскалации привилегий

CVE-2024-6387:
Уязвимость в OpenSSH, позволяющая неаутентифицированному удаленному пользователю получить доступ к системе через гонку состояний

CVE-2024-7646:
Уязвимость в ingress-nginx, позволяющая обойти проверку аннотаций и потенциально получить доступ к секретам кластера

CVE-2024-9042:
Уязвимость в Kubernetes, позволяющая выполнить произвольные команды на Windows-нодах через API-запросы к логам. Akamai

2024 Kubernetes Benchmark Report: основные выводы

Эффективность и надежность

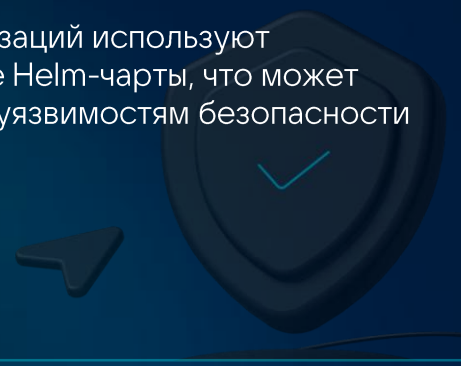
- 330 000+ рабочих нагрузок проанализировано в 2024 году
- 65% организаций не используют liveness/readiness-пробы, что снижает надёжность приложений
- 55% организаций имеют более 21% рабочих нагрузок без настроенных реплик, что увеличивает риск сбоев
- 67% организаций не указывают CPU-запросы для более 11% подов, что может привести к нестабильной работе кластеров

Оптимизация затрат

- 37% организаций имеют более 50% рабочих нагрузок, требующих настройки ресурсов для повышения эффективности
- 57% организаций уже сократили количество таких нагрузок до 10% или менее, внедряя практики rightsizing

Безопасность

- 28% организаций имеют более 90% рабочих нагрузок с небезопасными возможностями, такими как:
  - Разрешение привилегий
  - Запись в файловую систему
  - Запуск от root-пользователя
- 70% организаций используют устаревшие Helm-чарты, что может привести к уязвимостям безопасности



tech

Ужесточение политик по умолчанию

Переход от PodSecurityPolicy к Pod Security Standards

- PodSecurityPolicy (PSP) устарела и удалена начиная с Kubernetes 1.25
- Взамен внедряются Pod Security Standards (PSS):
  - Встроенные уровни: privileged, baseline, restricted
  - Используются через аннотации namespace'ов
  - Обеспечивают простую настройку безопасности без сторонних решений

Ограничения на работу контейнеров

Теперь все чаще:

- Запрет запуска контейнеров от root-пользователя (runAsNonRoot: true)
- Ограничение доступа к файловой системе: readOnlyRootFilesystem: true
- Ограничение привилегий: allowPrivilegeEscalation: false

Почему это важно

- Меньше поверхность атаки
- Упрощённый аудит и соответствие требованиям
- Повышение базового уровня безопасности без глубоких знаний у DevOps-команд

RBAC и принцип наименьших привилегий

Что меняется

Kubernetes теперь строго требует настройки Role-Based Access Control (RBAC)

По умолчанию роли и разрешения не выдаются автоматически — нужно настраивать явно

Расширено использование ServiceAccount'ов вместо привязки к пользователям

Почему это важно

Защищает кластер от эскалации привилегий и случайного доступа

Позволяет разделять доступ между разработкой, безопасностью и эксплуатацией

Упрощает аудит действий и соответствие требованиям (например, SOC 2, ISO 27001)

Интеграция с внешними системами безопасности

Что меняется

Kubernetes теперь часто интегрируется с внешними политиками и IDP:

- Open Policy Agent (OPA), Kyverno, Gatekeeper — для декларативного управления политиками
- SSO через OIDC, LDAP или Active Directory — для контроля аутентификации

Почему это важно

Централизация управления политиками и пользователями

Гибкий контроль доступа на уровне запросов и ресурсов

Снижение вероятности «дрейфа конфигураций» и ручных ошибок

Сканирование образов и контроль уязвимостей

Что меняется

Широкое распространение инструментов статического анализа контейнеров:
Trivy, Clair, Anchore — проверка на CVE, устаревшие пакеты, слабые конфигурации

Политики на уровне admission контроллеров блокируют уязвимые образы при деплое

Почему это важно

Уязвимости в образах — один из самых распространенных векторов атак

Позволяет устранять проблемы до попадания в продакшн

Упрощает соответствие стандартам безопасности (например, CIS Benchmarks)

Сетевая безопасность и Network Policies

Что меняется

Ранее все поды могли взаимодействовать между собой без ограничений

Теперь включаются NetworkPolicies или решения вроде Cilium, Calico

Часто используют Zero Trust подход — явное разрешение трафика

Почему это важно

Снижает риск распространения атаки между подами (lateral movement)

Позволяет создавать «белые списки» разрешенных коммуникаций

Ключевой элемент при построении микросервисной архитектуры

Внедрение DevSecOps и автоматизация безопасности

Что меняется

Безопасность теперь становится частью CI/CD:

- Проверки политик и сканирование образов — прямо в pipeline.
- Использование git-репозиториев как единого источника правды (GitOps).

Почему это важно

Обнаружение уязвимостей и ошибок на ранних этапах разработки

Повышает ответственность разработчиков за безопасность

Ускоряет релизы, снижает стоимость исправлений

Вывод: Безопасность в Kubernetes — это не опция, а необходимость

- Kubernetes стал зрелой платформой, и вместе с этим выросли требования к безопасности
- Автоматизация и стандартизация — ключ к снижению рисков и повышению устойчивости кластеров
- Инфраструктура и безопасность теперь неразделимы: DevOps → DevSecOps

Успешная защита кластера требует:

- Строгих политик по умолчанию, явного управления доступами
- Интеграции с внешними системами и проверками, глубокой сетевой сегментации
- И, главное — культуры ответственности за безопасность на всех этапах

Безопасный кластер — это не просто настройки. Это стратегия

tech

Процесс защиты контейнеров Kubernetes: от образов контейнеров до кластера и сети

tech

Защита на уровне контейнеров и образов

Цель: устранить риски до развертывания

Сканирование образов на уязвимости (Vulnerability Scanning)

Используются инструменты вроде Trivy, Grype, Clair для анализа Docker-образов на известные уязвимости (CVE). Интеграция в CI/CD позволяет «ловить» угрозы еще до деплоя

Подпись образов (Image Signing)

С помощью Cosign или Notary v2 можно подписывать и верифицировать образы, чтобы убедиться в их подлинности и целостности

Минималистичные образы (Minimal Base Images)

Использование образов вроде Alpine, Distroless — они содержат минимум пакетов и, соответственно, меньше потенциальных уязвимостей

Политики при приеме (Admission Policies)

Kubernetes может отклонять развертывания, если образ:

- не подписан
- содержит уязвимости
- не соответствует политике безопасности (например, через Kyverno или OPA Gatekeeper)

Вывод: контейнер — не черный ящик. Его нужно проверять как обычный код

Защита на уровне кластеров и подов

Цель: контроль среды исполнения и ограничение привилегий

RBAC (Role-Based Access Control)

Позволяет управлять, кто и что может делать в кластере (чтение, запись, администрирование). Настройка принципа наименьших привилегий — критична

Ограничение прав в контейнере

- Запрет запуска от root (runAsNonRoot)
- Отключение привилегий (allowPrivilegeEscalation: false)
- Только чтение (readOnlyRootFilesystem)
- Защита через seccomp и AppArmor профили

Pod Security Standards (PSS)

Встроенные профили безопасности:

- privileged: минимальные ограничения (нежелательно)
- baseline: безопасный минимум
- restricted: строгие ограничения

Заменили устаревшую PodSecurityPolicy

Изоляция namespace'ов и ресурсов

Разделение ресурсов между окружениями (dev, staging, prod), контроль квот и лимитов

Вывод: безопасность подов — это не только код, но и правильная настройка среды

Защита на уровне сети и трафика

Цель: исключить несанкционированное взаимодействие и утечку данных

Network Policies

Kubernetes позволяет настраивать, какие поды могут общаться друг с другом.

Без политики — по умолчанию разрешено все

NetworkPolicy — основа Zero Trust в кластере

Cilium и eBPF

Использование eBPF (в ядре Linux) для глубокой фильтрации сетевых запросов на уровне ядра:

- Высокая производительность
- Контроль DNS, egress-трафика, соединений на уровне L7 (HTTP)

Сервисная сетка (Service Mesh)

Решения вроде Istio, Linkerd добавляют:

- mTLS — автоматическое шифрование трафика между сервисами
- Аутентификация и авторизация,
- Аудит запросов и управление маршрутизацией

Ограничение исходящего трафика (Egress Control)

Часто упускаемый момент: поды могут «позвонить в интернет». Контролировать egress — важно для предотвращения утечек

Вывод: если атакующий попал в под — не дай ему уйти дальше

Как настраивать контроль доступа для пользователей и отслеживать подозрительные активности

tech

Контроль доступа и отслеживание активности в Kubernetes

Контроль доступа пользователей

- **RBAC (Role-Based Access Control):**
  Гибкое управление разрешениями. Используем Role и RoleBinding для namespace, ClusterRole — для всего кластера. Принцип наименьших привилегий — в приоритете
- **ServiceAccounts:**
  Для подов, автоматизации и CI/CD. Каждый компонент — отдельная учётка с ограниченными правами
- **Namespaces и изоляция:**
  Разделяем ресурсы по окружениям (dev, test, prod) и командам. Используем квоты и политики доступа на уровне namespace
- **SSO / OIDC / LDAP интеграция:**
  Внешняя аутентификация — централизованный контроль доступа, единый вход через корпоративные аккаунты

Отслеживание активности и инцидентов

- **Аудит-логи Kubernetes:**
  Все действия в API-сервере логируются (создание/удаление/доступ к ресурсам). Направляем в ELK, Loki, SIEM для анализа
- **Мониторинг API и событий:**
  Повышенное количество ошибок доступа (401, 403), аномальная активность — сигналы возможных атак.
- **Поведенческий анализ в рантайме:**
  Инструменты вроде Falco отслеживают системные вызовы в контейнерах: попытки exec, запись в /etc, доступ к сокетам и ядру
- **Интеграция с системами уведомлений:**
  Настройка алертов: Slack, Email, PagerDuty. Автоматическая реакция на события безопасности.

Вывод: если атакующий попал в под — не дай ему уйти дальше

Мониторинг метрик для реагирования на инциденты

tech

Контроль доступа и отслеживание активности в Kubernetes

Что мониторим:

- Системные метрики: загрузка CPU, память, диск, сеть (узлы и поды)
- Kubernetes-метрики: статус подов, количество перезапусков, задержки деплоя
- Приложенческие метрики: ошибки, время отклика, количество запросов (через Prometheus-интеграции)
- Инфраструктура и кластер: состояние etcd, scheduler, API-сервер

Как настроить:

- Prometheus + kube-state-metrics — стандартный стек для сбора и хранения метрик
- Grafana — дашборды с визуализацией, триггерами и алертами
- Node Exporter / cAdvisor — системные метрики с узлов и контейнеров
- Custom metrics — интеграция с приложениями (через Prometheus SDK или экспортеры)

Реакция на инциденты:

- Настройка алертов в Alertmanager по критическим порогам:
  - Подов нет в Ready
  - Частые перезапуски, API-сервер отвечает медленно
  - Высокий CPU на ноде
- Интеграция с Slack, Email, PagerDuty — мгновенные уведомления
- Визуализация трендов и аномалий для проактивного реагирования

Защита контейнеров

tech

Защита контейнеров

Shift Left	Kubernetes Security Posture Mgmt. (KSPM)	Cloud Workload Protection (CWPP)
Доверие своему коду	Безопасные настройки	Анализ поведения контейнеров
Сканирование образов	Контроль инфраструктуры	Контроль запущенных приложений

Видимость проблем и угроз

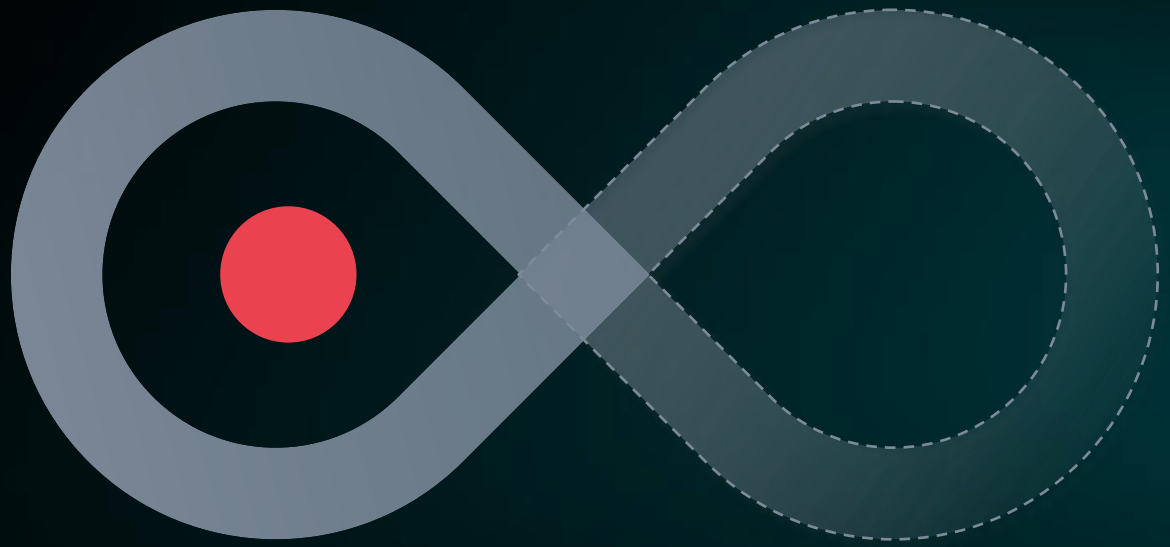
Снижение риска и контроль

Контроль Runtime

Защита Runtime

tech

Сканирование образов

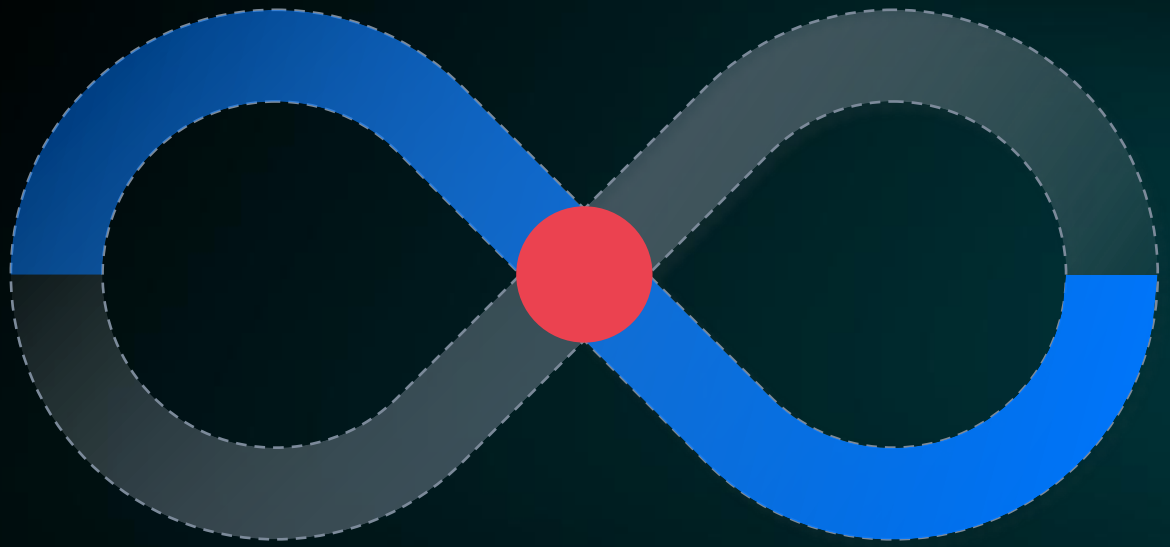


Комплексное и точное сканирование:

- Уязвимости
- Некорректные настройки и права доступа, пароли
- Вредоносное ПО

tech

Контроль инфраструктуры



Kubernetes Security Posture Management (KSPM):

- Сотни проверок на соответствие лучшим практикам
- Контроль соответствия CIS benchmarks

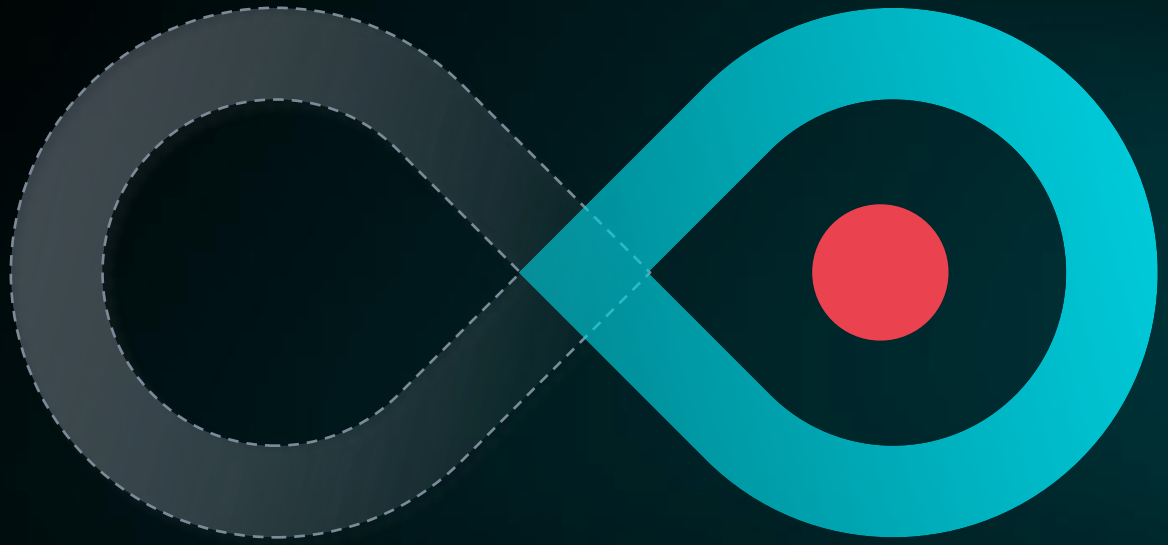
Визуализация угроз в Risk Explorer

Kube-enforcer admission controller

для контроля политик и прав доступа

tech

Контроль запущенных приложений



Контроль запускаемых images:

- Блокировка попыток запуска непроверенных образов

Блокировка вредоносной активности:

- Drift Prevention – блокировка запуска исполняемых компонентов
- Профилирование приложений и поведенческий анализ
- Виртуальный патчинг vShield
- CNDR (Cloud Native Detection & Response)

Микросегментация контейнеров в любой среде:

- L3-L4 Container Firewall

VK Cloud

Станислав Погоржельский,
технологический евангелист
VK Cloud

Спасибо за внимание!

tech