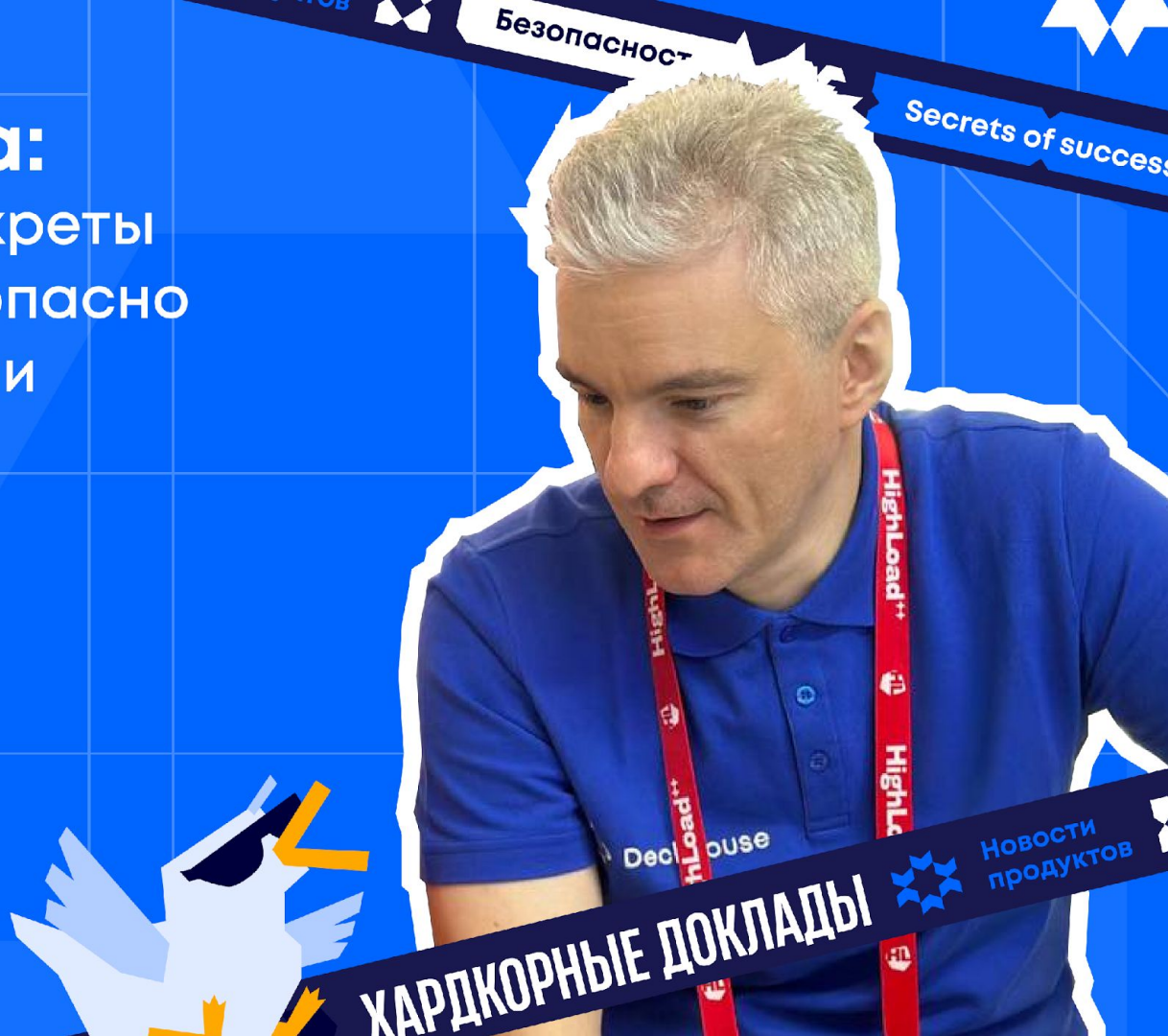
Успех секрета:
как доставлять секреты в приложения безопасно и без головной боли

Максим Киселев

Руководитель разработки
Deckhouse Stronghold



О ЧЁМ ДОКЛАД

Кто такие «секреты» и где они обитают

Чего нам не хватало в Vault

Как безопасно доставить секрет в приложение

Сколько проблем нас ждёт впереди

ЗАЧЕМ ВООБЩЕ СЕКРЕТЫ

- 80 % секретов нужны для аутентификации сервисов

ТОКЕНЫ API

ДОСТУПЫ К БД

ЗАЧЕМ ВООБЩЕ СЕКРЕТЫ

- 80 % секретов нужны для аутентификации сервисов
- 10 % — для шифрования/дешифрования данных или подписи сертификатов

ЗАЧЕМ ВООБЩЕ СЕКРЕТЫ

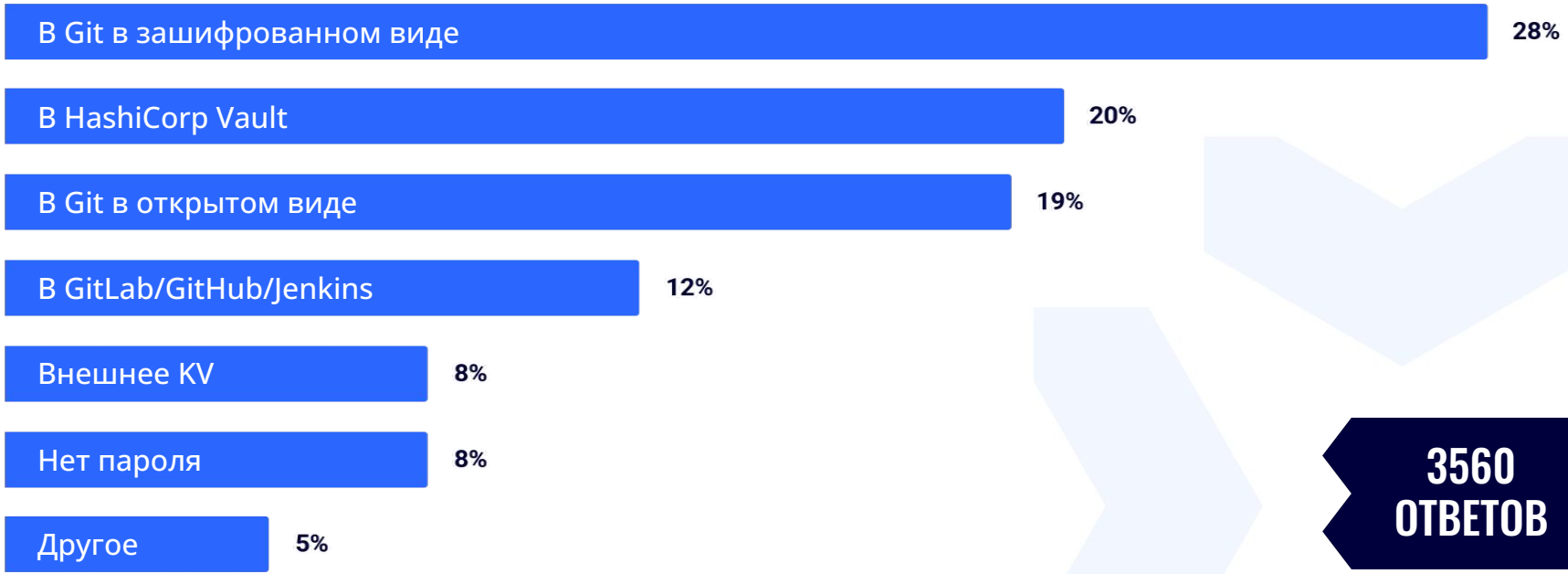
- 80 % секретов нужны для аутентификации сервисов
- 10 % — для шифрования/дешифрования данных или подписи сертификатов
- 10 % секретов — это вообще не секреты

АДРЕСА ХОСТОВ

НАЗВАНИЕ ОКРУЖЕНИЯ

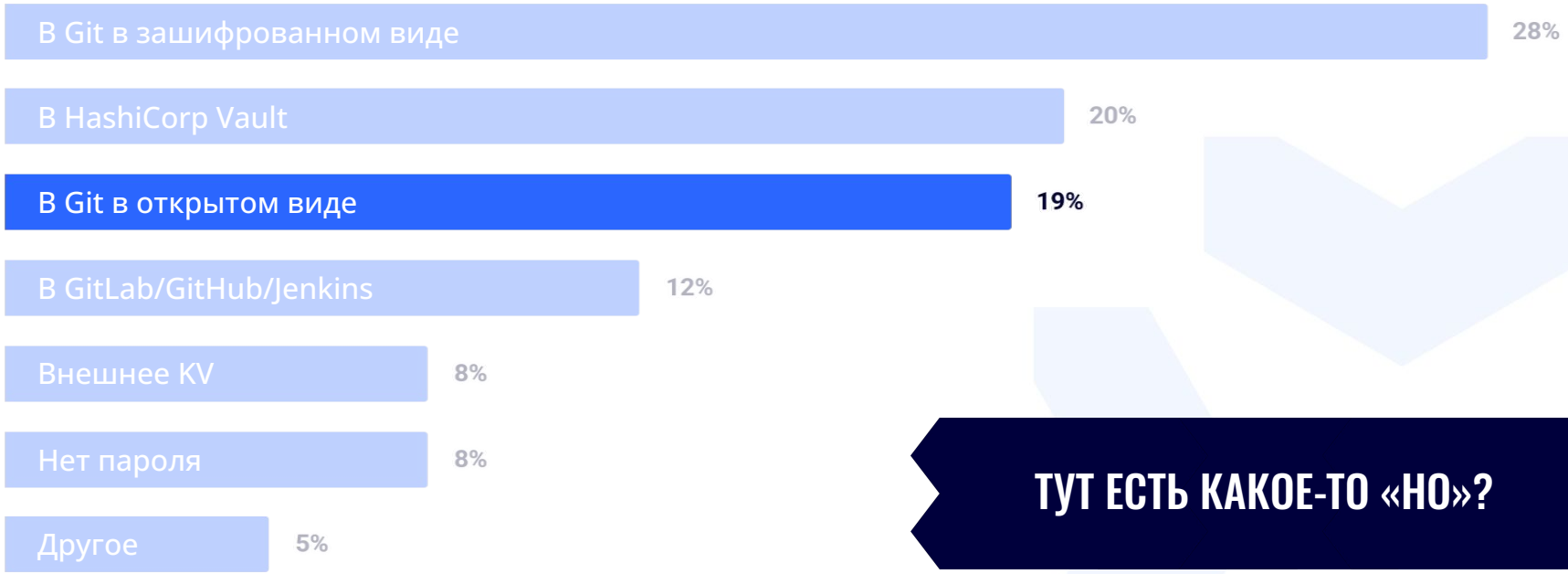
НАЗВАНИЯ МЕТОДОВ API

ГДЕ ВЫ ХРАНИТЕ ПАРОЛЬ ОТ БД?



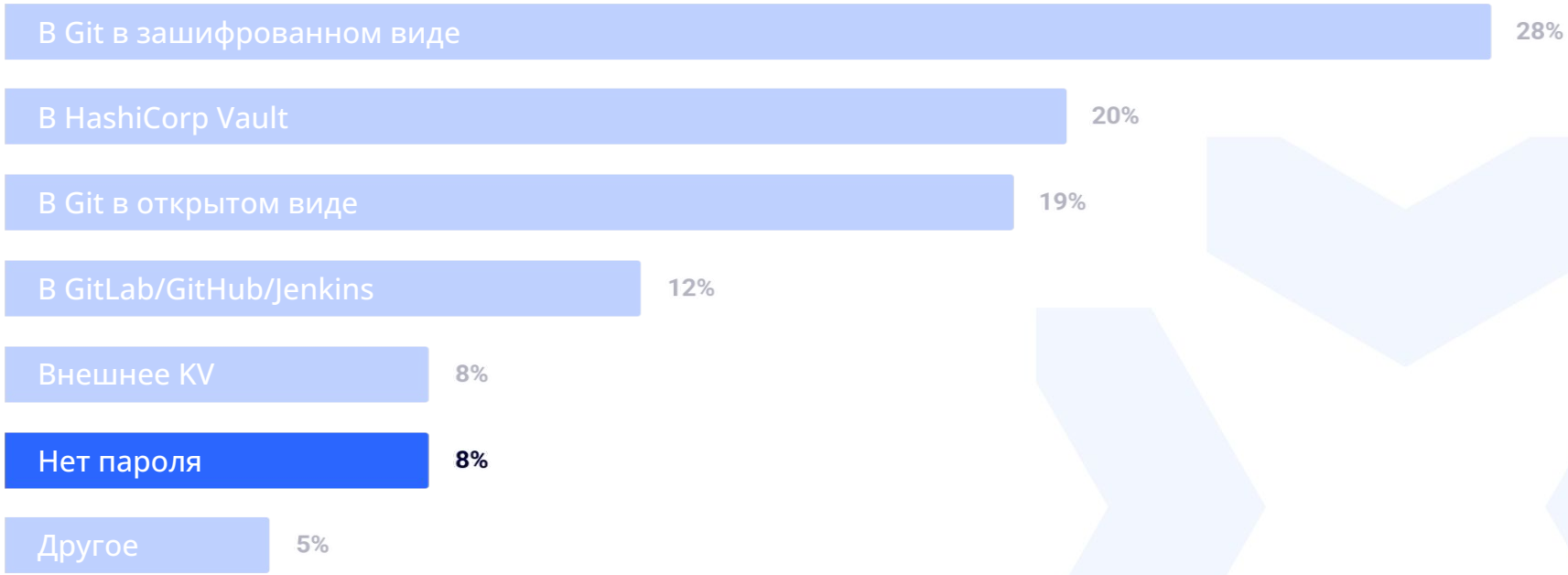
3560 ОТВЕТОВ

ГДЕ ВЫ ХРАНИТЕ ПАРОЛЬ ОТ БД?

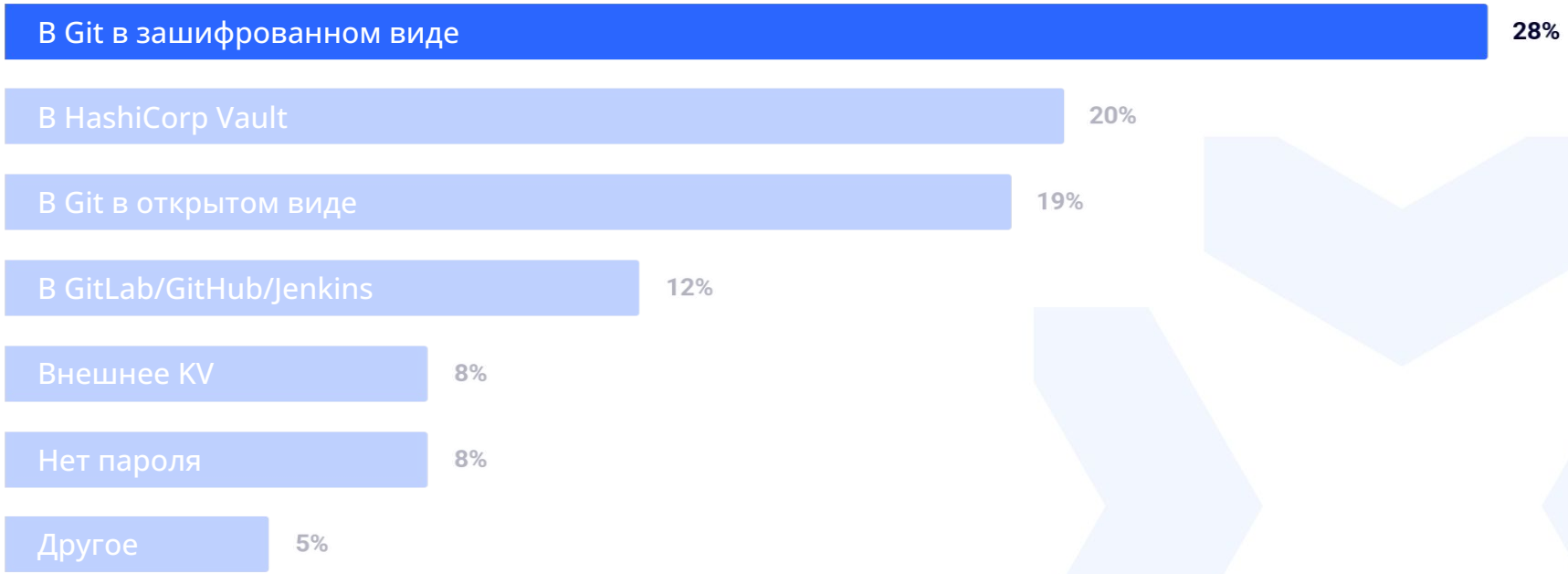


ТУТ ЕСТЬ КАКОЕ-ТО «НО»?

ГДЕ ВЫ ХРАНИТЕ ПАРОЛЬ ОТ БД?



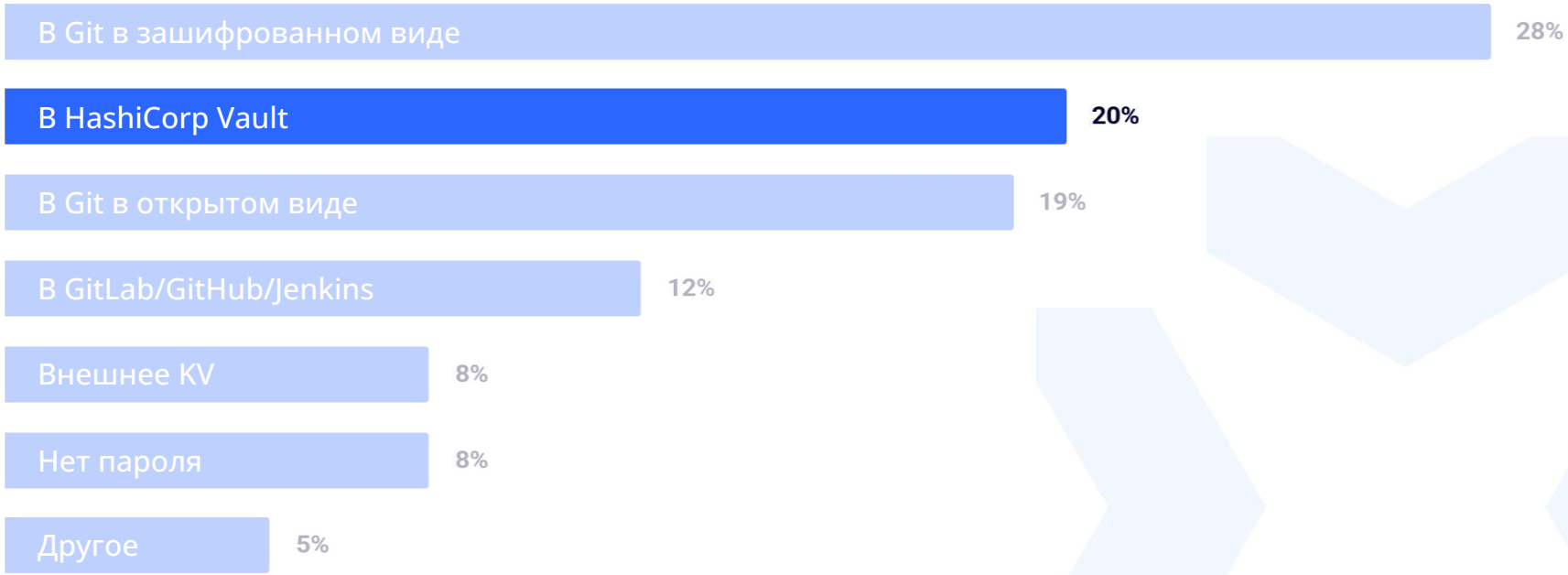
ГДЕ ВЫ ХРАНИТЕ ПАРОЛЬ ОТ БД?



В GIT В ЗАШИФРОВАННОМ ВИДЕ — 28%

- Если Git зашифрован,
  то где-то есть ключи расшифровки?
- Для разных проектов — разные
  ключи шифрования?
- Для разных секретов проекта —
  разные ключи шифрования?
- Может ли быть общий секрет
  для двух проектов?
- Изменить секрет — запускать
  CI/CD pipeline?
- Как провести ротейт секрета?
- Точно ли значение секрета
  для аутентификации — это часть
  конфигурации сервиса?

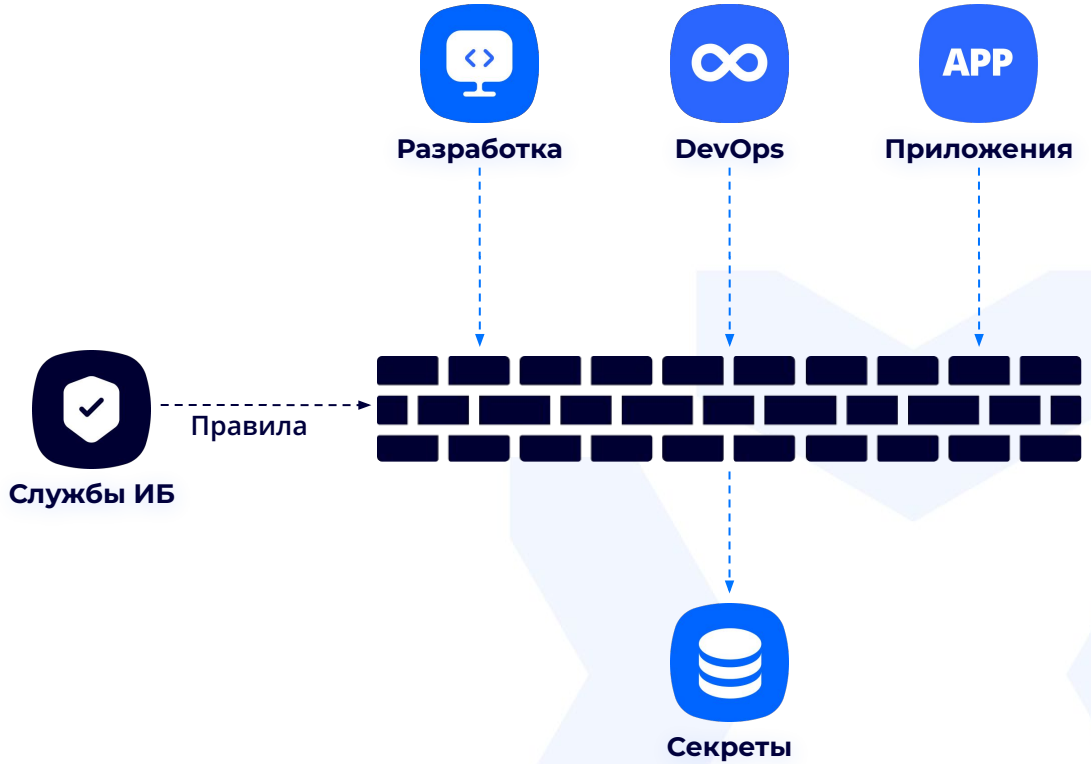
ГДЕ ВЫ ХРАНИТЕ ПАРОЛЬ ОТ БД?



ЗА

- Специально создан для хранения секретов
- Данные в хранилище зашифрованы
- Подходит и для людей, и для скриптов
- Можно разделить роли по выделению доступов к секретам и сам доступ к секретам

РАЗДЕЛИТЬ РОЛИ



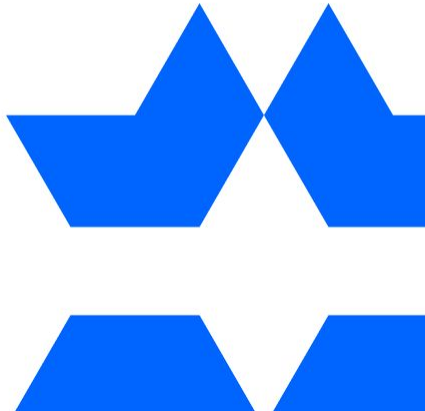
ЗА

- Специально создан для хранения секретов
- Данные в хранилище зашифрованы
- Подходит и для людей, и для скриптов
- Можно разделить роли по выделению доступов к секретам и сам доступ к секретам

ПРОТИВ

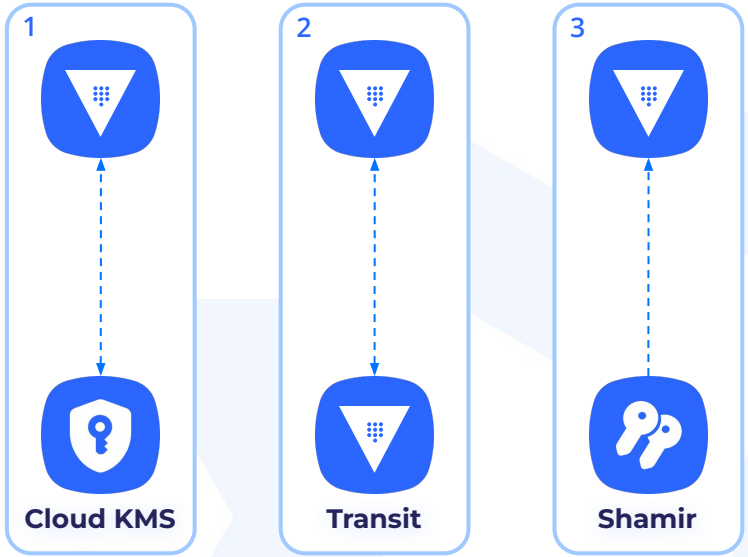
- Чтобы запустить Vault, нужно предоставить секрет
- Чтобы получить секрет из Vault, нужно предоставить ещё один секрет
- Доступ к секрету через API вместо ENV или File

РАСПЕЧАТАТЬ VAULT



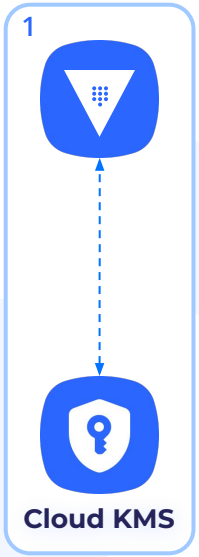
UNSEAL B VAULT

- Vault шифрует данные с помощью Encryption Key
- Encryption Key зашифрован, и ключ шифрования не хранится в Vault



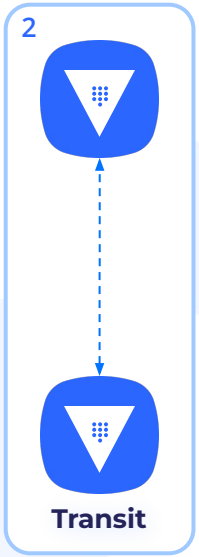
CLOUD KMS

- Не подходит для закрытых контуров
- Не подходит, если Vault планируется использовать как KMS



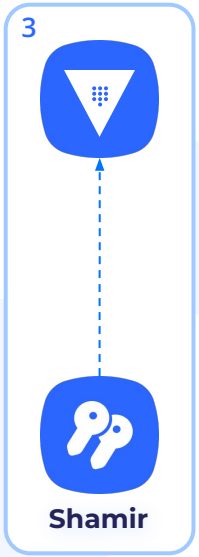
TRANSIT VAULT

- Как распечатывать транзитный Vault
- Если транзитный Vault утерян, данные не восстановить



РУЧНОЙ ВВОД КЛЮЧЕЙ

- Каждый перезапуск требует ручной распечатки
- Необходимо удостовериться, что ключи вводятся в легитимный Vault

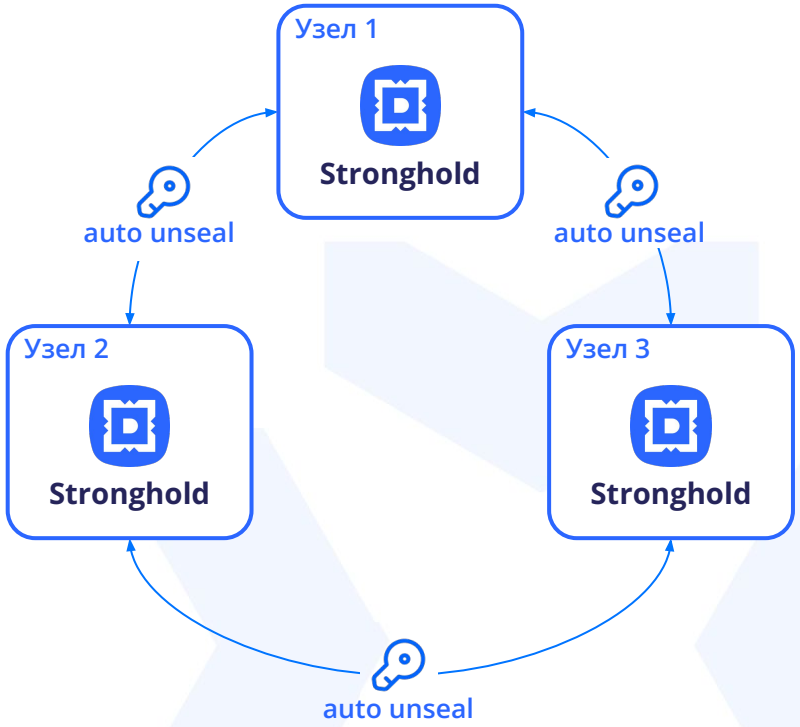


КАК РАСПЕЧАТАТЬ VAULT

- Когда нет KMS
- Нет другого Vault
- А вы всё равно хотите отказоустойчивость

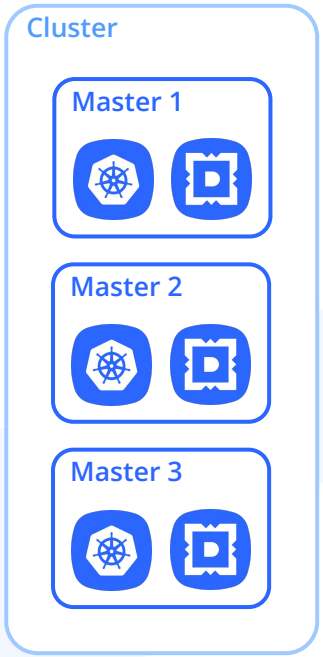
UNSEAL В STRONGHOLD

- Запускается внутри платформы из проверенных образов
- Автоматически инициализируется и хранит ключи в памяти
- Другие узлы кластера распечатываются в случае перезапуска и тоже могут выполнять распечатку
- Сохраняется возможность ручной распечатки при холодном старте



НАДЕЖНОСТЬ В STRONGHOLD

- Запускается на **Control-Plane**-узлах
- Если работает **API** – работает и **Stronghold**
- Расширенный функционал **RBAC** и аудита по сравнению с секретами в **API Kubernetes**
- Доступность **Kubernetes-секрета** и **Stronghold-секрета** эквивалентна



АУТЕНТИФИКАЦИЯ В VAULT

ПОЛУЧИТЬ СЕКРЕТ ИЗ VAULT

- В Kubernetes секрет можно подключить в контейнер, если можешь создать под в пространстве имён
- Чтобы получить секрет из Vault, нужен токен
- Чтобы получить токен, нужно аутентифицироваться

ПОХОЖЕ, У НАС LOOP?

В KUBERNETES У ПРИЛОЖЕНИЯ ЕСТЬ JWT

- Подключается в под как файл
- Имеет срок жизни (TTL)
- Однозначно идентифицирует под в кластере
- JWT можно провалидировать
- Обычно это Service Account пода,
  но не обязательно

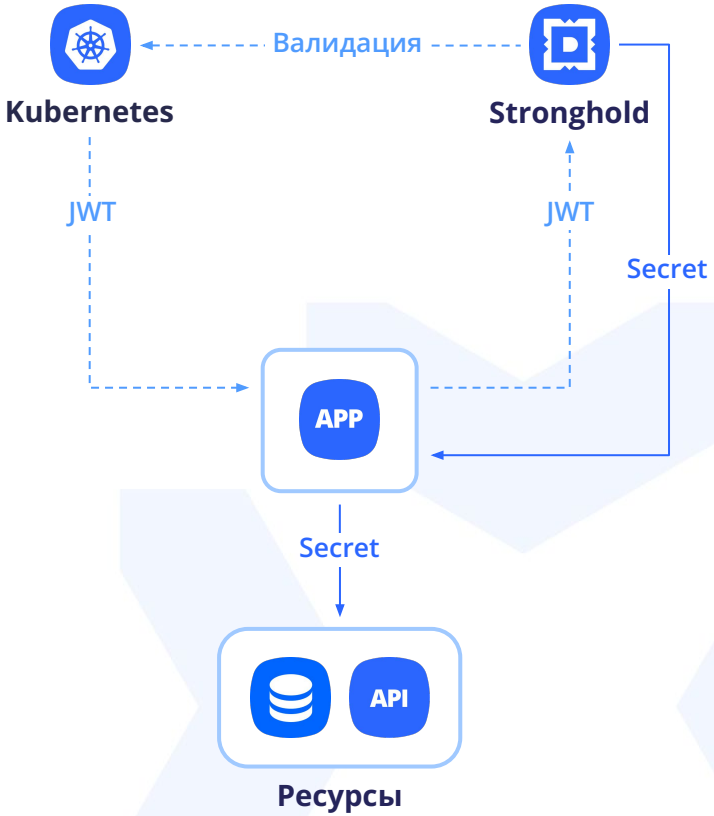
В KUBERNETES У ПРИЛОЖЕНИЯ ЕСТЬ JWT

- Подключается в под как файл
- Имеет срок жизни (TTL)
- Однозначно идентифицирует под
- JWT можно провалидировать
- Обычно это Service Account пода, но не обязательно


```
{
  "aud": [
    "https://kubernetes.default.svc.cluster.local"
  ],
  "exp": 1772211588,
  "iat": 1740675588,
  "iss": "https://kubernetes.default.svc.cluster.local",
  "kubernetes.io": {
    "namespace": "myapp-ns",
    "pod": {
      "name": "app",
      "uid": "f03fa951-25f0-45fe-8cc4-fac8077116c6"
    },
    "serviceaccount": {
      "name": "myapp",
      "uid": "f8379f63-e7ce-4069-90bc-933da862c3e3"
    },
    "warnafter": 1740679195
  },
  "nbf": 1740675588,
  "sub": "system:serviceaccount:myapp-ns:myapp"
}
```

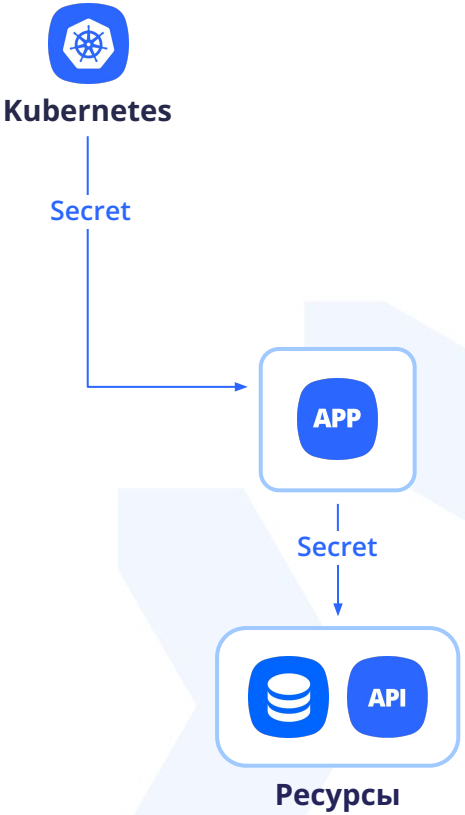
ПОЛУЧАЕМ СЕКРЕТ ИЗ STRONGHOLD

Выглядит как-то сложно, но безопасно



А ТАК БЫЛО В KUBERNETES

Выглядит просто, но откуда секрет в Kubernetes?



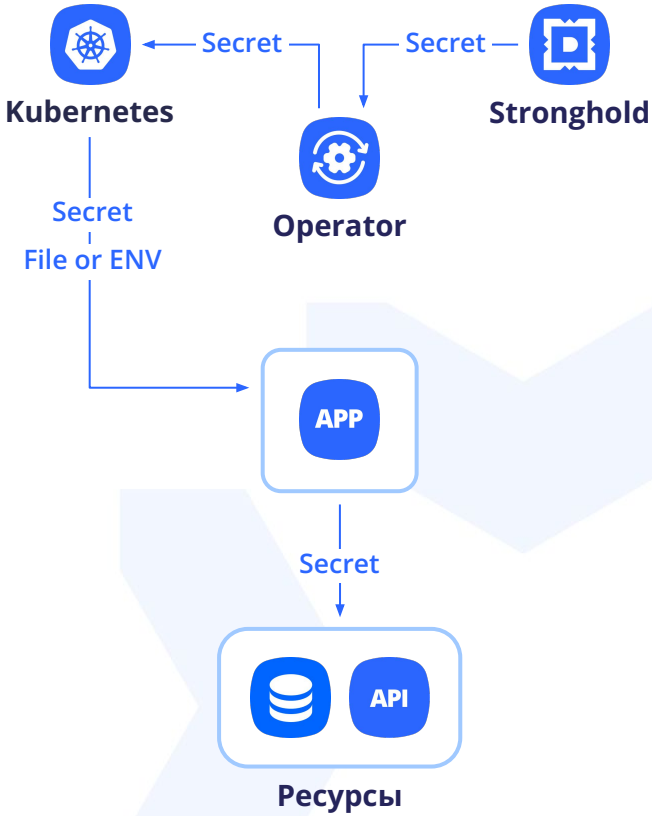
ХОЧУ В ПРИЛОЖЕНИИ ENV ИЛИ FILE

ДОСТАВЛЯЕМ СЕКРЕТЫ ИЗ VAULT/STRONGHOLD В KUBERNETES

- HASHICORP VAULT SECRETS OPERATOR
- EXTERNAL SECRETS OPERATOR
- HASHICORP VAULT AGENT INJECTOR
- VAULT SECRETS WEBHOOK
- HASHICORP VAULT CSI PROVIDER

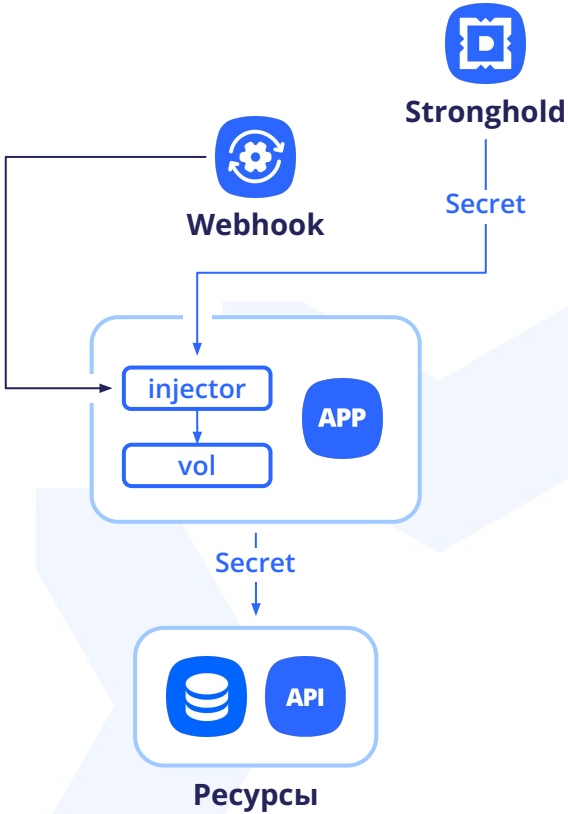
VAULT SECRETS OPERATOR И EXTERNAL SECRETS OPERATOR

- Позволяют синхронизировать ваши секреты из Vault в секреты Kubernetes
- Чарты для деплоя приложений не нужно менять
- Доступ к секретам можно реализовать через ServiceAccount Kubernetes
- Секреты доступны через API Kubernetes. Утерян аудит
- Секреты хранятся в etcd
- Секрет из Vault читает контейнер оператора, а не приложения, утерян аудит
- Другие приложения в пространстве имён могут подключить секрет как Volume



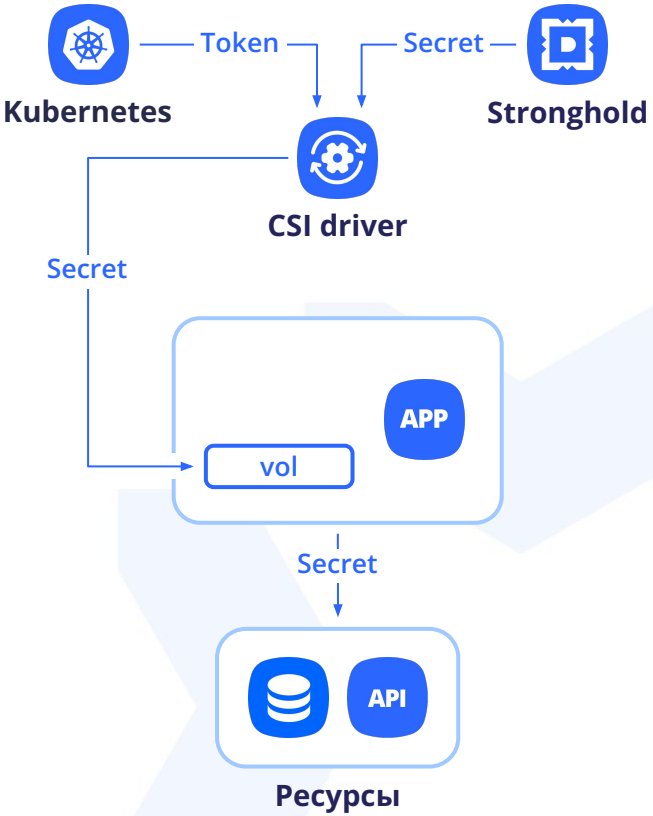
VAULT AGENT INJECTOR — ДОСТАВЛЯЕТ СЕКРЕТЫ В КОНТЕЙНЕР

- Секрет запрашивается напрямую из пода
- Для доступа к секретам используется Service Account приложения
- Секреты доставляются в приложение в виде файлов в Volume
- Секреты периодически обновляются агентом
- Агент постоянно запущен и потребляет ресурсы



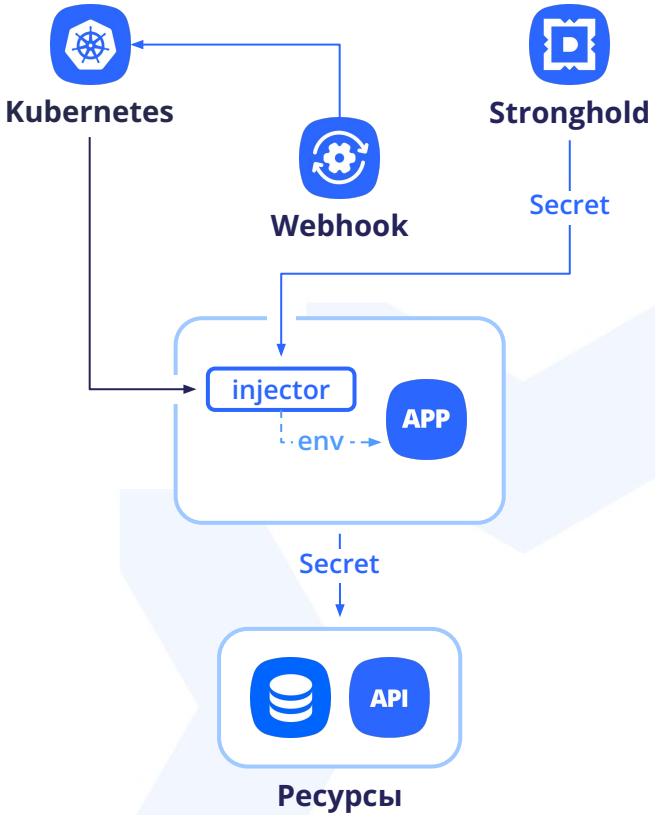
VAULT CSI PROVIDER

- Для доступа к секретам используется Service Account приложения
- Секрет не доступен в API Kubernetes
- Извлекает секрет из Vault в момент создания пода
- Секрет доступен в виде файла в контейнере
- Из пода приложения не требуется доступ к Vault



VAULT SECRETS WEBHOOK

- Для доступа к секретам используется Service Account приложения
- Секрет не доступен в API Kubernetes
- Извлекает секрет из Vault в момент создания пода
- Секрет доступен в виде ENV
- Инжектор после получения секрета заменяется процессом приложения



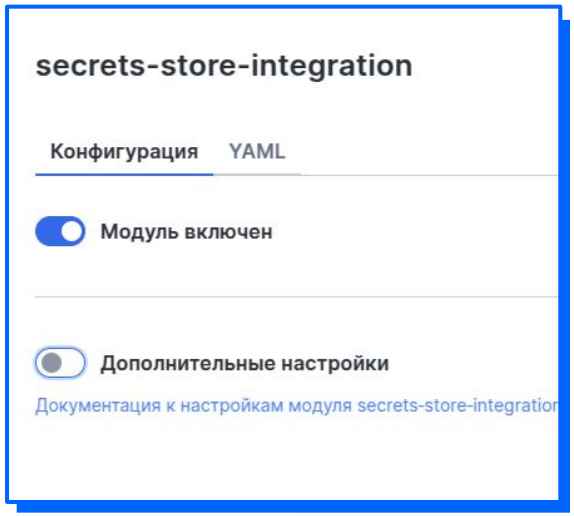
ДОСТАВЛЯЕМ СЕКРЕТЫ ИЗ VAULT/STRONGHOLD В KUBERNETES

- HASHICORP VAULT SECRETS OPERATOR
- EXTERNAL SECRETS OPERATOR
- HASHICORP VAULT AGENT INJECTOR
- VAULT SECRETS WEBHOOK
- HASHICORP VAULT CSI PROVIDER

ДОСТАВЛЯЕМ СЕКРЕТЫ ИЗ VAULT/STRONGHOLD В KUBERNETES

- HASHICORP VAULT SECRETS OPERATOR
- EXTERNAL SECRETS OPERATOR
- HASHICORP VAULT AGENT INJECTOR
- VAULT SECRETS WEBHOOK
- HASHICORP VAULT CSI PROVIDER

НАСТРОЙКА В DECKHOUSE




```
kind: Pod
apiVersion: v1
metadata:
 name: myapp
 namespace: myapp-namespace
 annotations:
   secrets-store.deckhouse.io/env-from-path:
                      secret/data/myapp-secrets
   secrets-store.deckhouse.io/role: myapp-role
spec:
 serviceAccountName: myapp
 containers:
   image: myapp:v1.0
   name: myapp
   command: ["/run/me"]
```

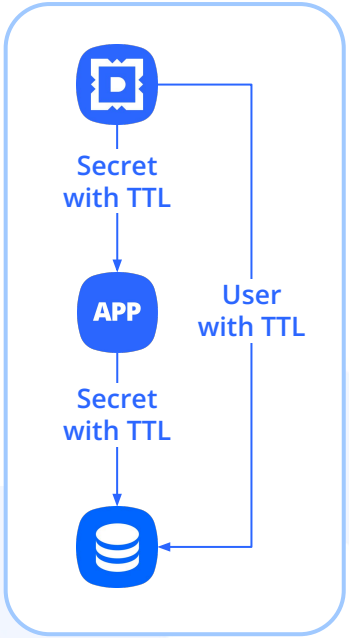
ЧТО ПОЛУЧИЛОСЬ?

- Доступами к секрету можно управлять в Vault независимо от RBAC Kubernetes
- Секрет хранится в Vault и извлекается из Vault
- Доступ к секрету может получить только определённый под

И ВСЁ?

ДИНАМИЧЕСКИЙ СЕКРЕТ

- В отличие от секретов Kubernetes секрет в Vault может быть динамическим, с коротким временем жизни, и создаваться в момент создания пода



НЕОБНОВЛЯЕМЫЙ ТОКЕН

- Используя необновляемый токен, можно запретить повторное получение секрета из Vault
- Но при перезапуске контейнера токен будет обновлён и секрет единожды получен

```
spec:
  automountServiceAccountToken: false
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
      - name: custom-token
        mountPath: /vault-token
        subPath: token
  volumes:
  - name: custom-token
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          path: token
          expirationSeconds: 600
          audience: stronghold
```

ДОСТУП К СЕКРЕТНЫМ ENV КОНТЕЙНЕРА

Переменные окружения, полученные из Vault, недоступны при exec в контейнер, но могут быть прочитаны из `/proc/1/environ`

- Приложение должно сделать `unset/clearenv`
- Приложение должно очистить `/proc/self/environ`
- Или вести аудит запросов с помощью **runtime-audit-engine,** используя фильтр

```
evt.type = openat and
fd.filename = environ
```

ОЧИСТИТЬ ENVIRON

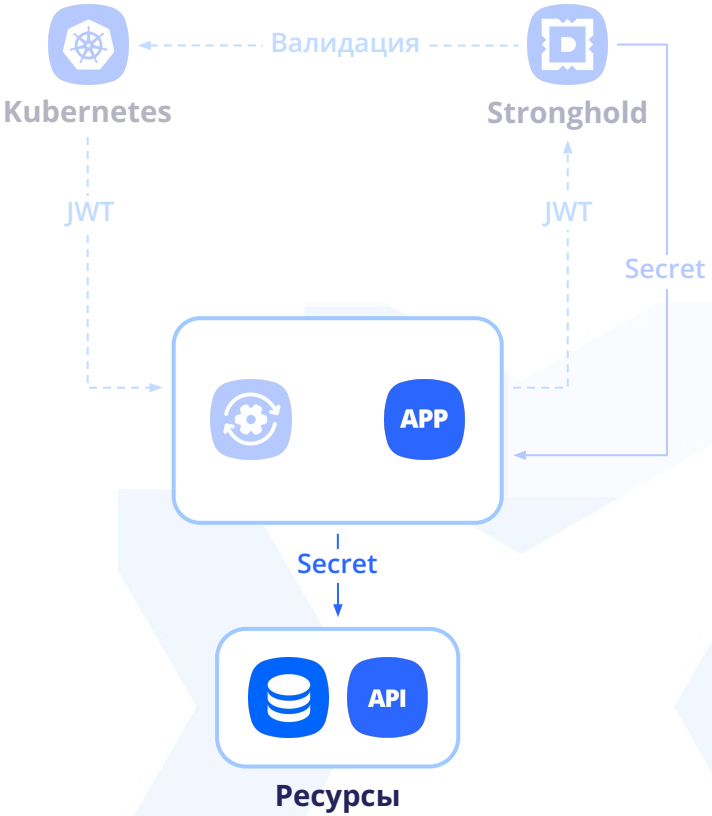
- В языке C environ — это массив указателей на строки
- При очистке этих строк очищается и содержимое файла /proc/self/environ

```
#include <unistd.h>
#include <string.h>

int main(int argc, char* argv[], char* envp[]) {
 while (*envp) {
   memset(*envp, 0, strlen(*envp));
   envp++;
 }
 sleep(60); // можно проверить, что в environ
пусто
 return 0;
}
```

БЕЗОПАСНОЕ ХРАНЕНИЕ И ДОСТАВКА СЕКРЕТОВ КАК ЧАСТЬ ПЛАТФОРМЫ

- Секреты хранятся в безопасном Stronghold
- Доступ к секрету по JWT пода
- Секреты не попадают в промежуточный слой Kubernetes
- Для приложения это привычные ENV или файлы
- При правильном подходе секреты недоступны даже внутри контейнера



КАКИЕ ПРОБЛЕМЫ НЕ РЕШЕНЫ ИЛИ РЕШЕНЫ ЧАСТИЧНО

ДОСТАВКА СЕКРЕТОВ — ПОЛОВИНА ДЕЛА

Ошибки в приложении

- Некорректная работа приложения может привести к утечкам секретов

Административный доступ к системе виртуализации

- Позволяет получить доступ к памяти виртуальных машин, в том числе запущенным там процессам приложений и Stronghold

Логи приложения

- Секреты могут попасть в логи приложения

Административный доступ на узел с приложением

- Позволит провести перехват секретов, в том числе через дамп памяти

Административный доступ на узел с Stronghold

- Позволит провести перехват ключей шифрования хранилища и расшифровать все данные

ХАРДКОРНЫЕ ДОКЛАДЫ Новости продуктов Безопасность Secrets of success

ОЦЕНИТЕ ДОКЛАД

Максима Киселева

Контакты спикера:

@trublast

ХАРДКОРНЫЕ ДОКЛАДЫ Новости продуктов