Методы аутентификации. OIDC провайдеры

В данной статье собраны основные шаги по настройке приложения OIDC для различных провайдеров. Для получения более общей информации об использовании и работе см. документацию по методу StarVault JWT/OIDC.

Провайдеры OIDC часто обладают широкими возможностями настройки, поэтому вам следует ознакомиться с их рекомендуемыми параметрами и передовым опытом. Руководства, перечисленные ниже, в основном составлены сообществом и призваны помочь вам начать работу. Исправления и дополнения могут быть представлены через репозиторий StarVault Github.

- Gitlab
- Keycloak
- Kubernetes

1. Gitlab

1. Перейдите в раздел **Настройки > Приложения**
2. Заполните имя и URL перенаправления
3. Убедитесь, что выбрали область **openid**
4. Скопируйте идентификатор и секрет клиента

2. Keycloak

1. Выберите/создайте пространство и клиента
2. Выберите клиента и перейдите в раздел **Настройки**
3. Проверьте настройки. Они должны соответствовать перечисленным ниже:
    - **Протокол клиента**: openid-connect
    - **Тип доступа**: конфиденциальный
    - **Стандартный поток**: включен
4. Настройте действительные URL перенаправления
5. Нажмите **Сохранить**

6. Перейдите к разделу **Учетные данные**
7. Выберите **Client ID** > **Secret** и запишите сгенерированный секрет

3. Kubernetes

Kubernetes может выступать в качестве OIDC-провайдера, чтобы StarVault мог подтверждать токены учетных записей служб с помощью аутентификации JWT/OIDC.

> Механизм JWT auth не использует API TokenReview от Kubernetes при аутентификации, а вместо этого использует криптографию с открытым ключом для проверки содержимого JWT. Это означает, что токены, которые были отозваны Kubernetes, будут считаться действительными в StarVault до истечения их срока действия. Чтобы снизить этот риск, используйте короткие TTL для токенов учетных записей сервисов или используйте Kubernetes auth, который использует TokenReview API.

3.1. Использование обнаружения эмитента учетной записи

При использовании обнаружения эмитента учетной записи нужно предоставить только аутентификацию JWT с монтированным URL-адресом обнаружения OIDC, и иногда центр сертификации TLS, которому можно доверять. Это делает данный метод наиболее простым в настройке, если кластер Kubernetes соответствует требованиям.

Требования к кластеру Kubernetes:

- Функция `ServiceAccountIssuerDiscovery` включена.
  - Присутствует с версии 1.18, по умолчанию включается с версии 1.20.
- Флаг `kube-apiserver --service-account-issuer` установлен на URL, доступный из StarVault. По умолчанию публичный для большинства управляемых решений Kubernetes.
- При входе в систему необходимо использовать недолговечные токены учетных записей сервисов.
  - Начиная с версии 1.21 токены, устанавливаемые в поды, по умолчанию короткоживущие.

Этапы настройки:

1. Убедитесь, что URL-адреса обнаружения OIDC не требуют аутентификации:

```bash
kubectl create clusterrolebinding oidc-reviewer  \
   --clusterrole=system:service-account-issuer-discovery \
   --group=system:unauthenticated
```

2. Найдите URL-адрес эмитента кластера.

```bash
ISSUER="$(kubectl get --raw /.well-known/openid-configuration | jq -r
'.issuer')"
```

3. Включите и настройте JWT-аутентификацию в StarVault.
4. Если StarVault работает в Kubernetes:

```bash
kubectl exec starvault-0 -- starvault auth enable jwt
kubectl exec starvault-0 -- starvault write auth/jwt/config \
   oidc_discovery_url=https://kubernetes.default.svc.cluster.local \

oidc_discovery_ca_pem=@/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
```

5. В качестве альтернативы, если StarVault не запущен в Kubernetes:

> Когда StarVault находится вне кластера, указанная ниже конечная точка $ISSUER может быть доступна или недоступна. Если это не так, можете настроить JWT-аутентификацию с помощью `jwt_validation_pubkeys`.

```bash
starvault auth enable jwt
starvault write auth/jwt/config oidc_discovery_url="${ISSUER}"
```

6. Настройте роль и войдите в систему, как описано ниже.

3.2. Использование открытых ключей с проверкой JWT

Этот метод может быть полезен, если API Kubernetes недоступен из StarVault или если хотите, чтобы одна смонтированная JWT-аутентификация обслуживала несколько кластеров Kubernetes, объединяя их публичные подписанные ключи.

Требования к кластеру Kubernetes:

- Функция `ServiceAccountIssuerDiscovery` включена.
  - Присутствует с версии 1.18, по умолчанию включается с версии 1.20.
- Этого требования можно избежать, если возможно получить доступ к главным узлам Kubernetes для чтения открытого ключа подписи непосредственно с диска по адресу `/etc/kubernetes/pki/sa.pub`. В этом случае можете пропустить шаги по получению и последующему преобразованию ключа, так как он уже будет в формате **PEM**.
- При входе в систему необходимо использовать недолговечные токены учетных записей сервисов.
  - Начиная с версии 1.21 токены, устанавливаемые в поды, по умолчанию короткоживущие.

Этапы настройки:

1. Получите открытый ключ подписи учетной записи службы из JWKS URI вашего кластера.

```bash
# Query the jwks_uri specified in /.well-known/openid-configuration
kubectl get --raw "$(kubectl get --raw /.well-known/openid-configuration | jq -r '.jwks_uri' | sed -r 's/.*\.[^/]+(.*)/\1/')"
```

2. Конвертируйте ключи из формата JWK в PEM. Вы можете использовать инструмент CLI или онлайн-конвертер.
3. Настройте монтирование JWT-аутентификации с полученным открытыми ключами.

```bash
starvault write auth/jwt/config \
   jwt_validation_pubkeys="-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9...
-----END PUBLIC KEY-----","-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9...
-----END PUBLIC KEY-----"
```

4. Настройте роль и войдите в систему, как описано ниже.

3.3. Создание роли и вход в систему

После того как JWT-аутентификатор настроен, можете настроить роль и войти в систему. Ниже предполагается, что используется токен учетной записи проектируемого сервиса, доступный во всех подах по умолчанию. Если хотите контролировать аудиторию или TTL, смотрите раздел Указание TTL и аудитории ниже.

1. Выберите любое значение из массива аудиторий по умолчанию. В этих примерах в массиве aud есть только одна аудитория — `https://kubernetes.default.svc.cluster.local`.

   Чтобы найти аудитории по умолчанию, либо создайте свежий токен (требуется `kubectl` v1.24.0+):

```bash
kubectl create token default | cut -f2 -d. | base64 --decode
```

   Или прочитайте токен из файловой системы запущенного пода:

```bash
kubectl exec my-pod -- cat /var/run/secrets/kubernetes.io/serviceaccount/token | cut -f2 -d. | base64 --decode
```

2. Создайте роль для JWT-аутентификации, которую может использовать сервисная учетная запись `default` из пространства имен `default`.

```bash
starvault write auth/jwt/role/my-role \
   role_type="jwt" \
   bound_audiences="<AUDIENCE-FROM-PREVIOUS-STEP>" \
   user_claim="sub" \
   bound_subject="system:serviceaccount:default:default" \
   policies="default" \
   ttl="1h"
```

3. После этого поды или другие клиенты, имеющие доступ к сервисной учетной записи JWT, могут войти в систему.

```bash
starvault write auth/jwt/login \
   role=my-role \
   jwt=@/var/run/secrets/kubernetes.io/serviceaccount/token
# ИЛИ эквивалентно:
curl \
   --fail \
   --request POST \
   --header "X-Vault-Request: true" \
   --data '{"jwt":"<JWT-TOKEN-HERE>","role":"my-role"}' \
   "${STARVAULT_ADDR}/v1/auth/jwt/login"
```

3.4. TTL и аудитория

Если хотите указать пользовательский TTL или аудиторию для токенов учетных записей сервисов, в следующей спецификации pod показано монтирование тома, которое отменяет встроенный токен по умолчанию. Это особенно актуально, если не можете отключить флаг `--service-account-extend-token-expiration` для `kube-apiserver` и хотите использовать короткие TTL.

При использовании полученного токена нужно будет установить значение `bound_audiences=starvault` при создании ролей в StarVault в монтированном JWT-аутентификаторе.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  # automountServiceAccountToken в этом примере избыточен, потому что
используемый
  # mountPath совпадает с default path. Это совпадение не даёт создаться
  # стандартному встроенному токену доступа. Вы можете использовать эту опцию,
```

```
  # чтобы быть уверенными, что только один токен будет примонтирован, если
используется другой путь монтирования
  automountServiceAccountToken: false
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
      - name: custom-token
        mountPath: /var/run/secrets/kubernetes.io/serviceaccount
  volumes:
  - name: custom-token
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          path: token
          expirationSeconds: 600 # 10 minutes is the minimum TTL
          audience: starvault        # Должен совпадать с `bound_audiences` JWT
роли
      # Оставшиеся sources включены, чтобы имитировать оставшиеся стандартные
внедряемые тома системы
      # контроля допуска
      - configMap:
          name: kube-root-ca.crt
          items:
          - key: ca.crt
            path: ca.crt
      - downwardAPI:
          items:
          - fieldRef:
              apiVersion: v1
              fieldPath: metadata.namespace
            path: namespace
```

Методы аутентификации. RADIUS

Метод `radius` auth позволяет пользователям аутентифицироваться в StarVault с помощью существующего сервера RADIUS, который принимает схему аутентификации PAP.

1. Аутентификация

Путь по умолчанию `- /radius`. Если этот метод аутентификации был включен для другого пути, укажите `-path=/my-path` в командной строке.

1.1. Через CLI

BASH

```
$ starvault login -method=radius username=sethvargo
```

1.2. Через API

Конечной точкой по умолчанию является `auth/radius/login`. Если этот метод аутентификации был включен по другому пути, используйте это значение вместо `radius`.

BASH

```
$ curl \
    --request POST \
    --data '{"password": "..."}' \
    http://127.0.0.1:8200/v1/auth/radius/login/sethvargo
```

Ответ будет содержать токен по адресу `auth.client_token`:

BASH

```
{
  "auth":
  {
    "client_token": "c4f280f6-fdb2-18eb-89d3-589e2e834cdb",
    "policies": ["admins"],
    "metadata":
    {
      "username": "mitchellh"
    }
  }
}
```

2. Конфигурация

2.1. Через CLI

1. Включите метод аутентификации `radius`:

   ```bash
   $ starvault auth enable radius
   ```

2. Настройте сведения о подключении к вашему RADIUS-серверу.

   ```bash
   $ starvault write auth/radius/users/mitchellh policies=admins
   ```

   Полный список параметров конфигурации приведен в документации по API.

   Вышеописанное создает новое сопоставление для пользователя "mitchellh", которое будет связано с политикой "администраторы".

   В качестве альтернативы, StarVault может назначить настраиваемый набор политик любому пользователю, который успешно прошел аутентификацию на сервере RADIUS, но не имеет явного соответствия в параметре `users/` path. Это делается с помощью параметра конфигурации `unregistered_user_policies`.

3. API

Метод RADIUS auth имеет полный HTTP API. Пожалуйста, ознакомьтесь с RADIUS Auth API для получения более подробной информации.

Методы аутентификации. Токен

Метод аутентификации **Token** является встроенным. Он автоматически доступен по пути **/auth/token**. Этот метод позволяет пользователям аутентифицироваться с использованием токена, а также создавать новые токены, отзывать секреты по токену и многое другое.

Когда любой другой метод аутентификации возвращает идентификатор, ядро StarVault вызывает метод **Token** для создания нового уникального токена для этого идентификатора.

Метод аутентификации **Token** также может быть использован для обхода любого другого метода аутентификации: вы можете напрямую создавать токены, а также выполнять другие операции с токенами, такие как продление и отзыв.

Для метода аутентификации **Token** недоступна функция встроенной мультифакторной ауатентификации (MFA).

1. Концепция токенов

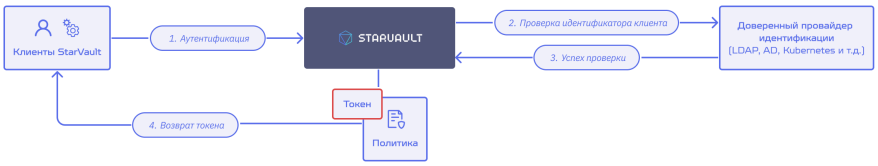
Токены являются основным методом аутентификации в StarVault.

Например, администратор StarVault входит в систему через метод аутентификации **Token**, используя начальный корневой токен, чтобы настроить другие методы аутентификации.

Токены можно использовать напрямую или динамически генерировать токены на основе внешних идентификаторов с помощью методов аутентификации.

Это первый метод аутентификации для StarVault. Это также единственный метод аутентификации, который нельзя отключить.

В StarVault токены сопоставляются с информацией. Самая важная информация, сопоставляемая с токеном, - это набор из одной или нескольких прикрепленных политик. Эти политики определяют, что владельцу токена разрешено делать в StarVault. Также сопоставляемая информация включает метаданные, которые можно просматривать и которые добавляются в журнал аудита, например время создания, время последнего обновления и т.д.



1.1. Типы токенов

Существует два типа токенов: **service** и **batch**.

1.1.1. Токены типа service

Сервисные токены — это то, что пользователи обычно воспринимают как "обычные" токены StarVault. Они поддерживают все функции, такие как продление, отзыв, создание дочерних токенов и многое другое. Соответственно, они требуют больших ресурсов для создания и отслеживания.

Аренды, созданные сервисными токенами (включая аренды дочерних токенов), отслеживаются вместе с токеном и аннулируются по истечении его срока действия.

1.1.2. Токены типа batch

Пакетные токены - это зашифрованные блобы (наборы двоичных данных), которые содержат достаточно информации для их использования в действиях с StarVault, но для их отслеживания не требуется хранение на диске. В результате они являются чрезвычайно легковесными и масштабируемыми, что делает пакетные токены идеальными для операций, где требуется высокая масштабируемость и минимальное использование ресурсов. Пакетные токены лишены большей части гибкости и функций сервисных токенов.

Аренды, созданные пакетными токенами, ограничены оставшимся временем жизни (TTL) пакетных токенов и, если пакетный токен не является токеном-сиротой (orphan), отслеживаются родительским токеном. Они аннулируются, когда истекает TTL пакетного токена, или когда отзывается родительский токен (в этот момент пакетный токен также теряет доступ к StarVault).

Как следствие, пакетные токены могут использоваться в кластерах с репликацией производительности, но только если они являются токенами-сиротами, поскольку не-сиротские токены не смогут обеспечить проверку действительности родительского токена.

1.1.3. Сравнение токенов типа batch и service

Эта справочная таблица описывает разницу в поведении сервисных и пакетных токенов.

	Токены service	**Токены batch**
Могут быть корневыми токенами	Да	Нет

	Токены service	Токены batch
Можно создавать дочерние токены	Да	Нет
Может быть возобновляемым	Да	Нет
Отзыв вручную	Да	Нет
Может быть периодическим	Да	Нет
Может иметь явное максимальное значение TTL	Да	Нет (всегда используется фиксированный срок жизни)
Имеет уникальный идентификатор, связанный с токеном (accessor)	Да	Нет
Имеет персональное хранилище (cubbyhole)	Да	Нет
Отзыв вместе с родителем (если не является сиротой)	Да	Перестает работать
Назначение аренды динамических секретов	Самостоятельно	Родитель (если не сирота)
Может использоваться в кластерах репликации	Нет	Да (если сирота)
Масштабирование создания с учетом производительности резервных узлов	Нет	Да
Расходы	Тяжелый вес; многократная запись в хранилище при создании токена	Легкий вес; нет затрат на хранение при создании токена

1.2. Корневые токены

Корневые токены - это токены, к которым прикреплена политика **root**. Корневые токены могут выполнять любые действия в StarVault. Кроме того, это единственный тип токенов в StarVault, который можно настроить так, чтобы срок его действия никогда не истекал и не требовал продления. В результате, создание корневых токенов намеренно затруднено; на самом деле существует только три способа создания корневых токенов:

1. **Использование существующего корневого токена**. Если у вас уже есть корневой токен, вы можете использовать его для создания нового корневого токена. Однако стоит отметить, что корневой токен с ограниченным сроком действия не может создать корневой токен, который не истекает.

2. **С помощью процесса генерации нового корневого токена**. Этот процесс используется для создания нового корневого токена без необходимости иметь существующий корневой токен. Он требует участия кворума владельцев ключей распечатывания (unseal keys) и обычно выполняется через команду `starvault operator generate-root`. Процесс является многошаговым и включает в себя начало процесса, предоставление частей ключа разблокировки от различных владельцев, и завершение процесса для получения нового корневого токена.
3. **При инициализации StarVault**. Когда вы впервые инициализируете StarVault, генерируется начальный корневой токен. Этот токен используется для первоначальной настройки StarVault, включая настройку других методов аутентификации и политик.

> Использование корневых токенов сопряжено с высокими рисками безопасности, так как они дают неограниченный доступ ко всему StarVault. По этой причине рекомендуется ограничить использование корневых токенов и, по возможности, полагаться на другие методы аутентификации и делегирования полномочий через политики и роли.

1.3. Иерархии токенов и токены-сироты

Обычно, когда владелец токена создает новые токены, эти токены создаются как дочерние по отношению к оригинальному токену. Когда родительский токен отзывается, все его дочерние токены — и все их аренды — также отзываются. Это гарантирует, что пользователь не может избежать отзыва токена, просто создавая бесконечное дерево дочерних токенов.

Часто такое поведение нежелательно, поэтому пользователи с соответствующим доступом могут создавать токены-сироты (**orphan**). У этих токенов нет родителя — они являются корнем собственного дерева токенов. Токены-сироты могут быть созданы:

- С помощью `write` в конечной точке **auth/token/create-orphan**.
- С помощью доступа sudo или root к **auth/token/create**, установив параметр `no_parent` в значение `true`.
- Через роли хранилища токенов.
- Через вход в систему с использованием любого другого (не токенного) метода аутентификации.

Пользователи с соответствующими правами могут также использовать конечную точку **auth/token/revoke-orphan**, которая отзывает заданный токен, но вместо того, чтобы отзывать остальные части дерева, она устанавливает ближайших детей токена в статус сирот.

1.4. Аксессоры токенов

При создании токена, также создается и возвращается **аксессор** токена. Этот аксессор является значением, которое действует как ссылка на токен и может использоваться только для выполнения ограниченного набора действий:

- Просмотр свойств токена (не включая фактический идентификатор токена)
- Просмотр возможностей токена на пути
- Продление токена
- Отзыв токена

Токен, осуществляющий вызов, должен иметь соответствующие разрешения для этих функций.

Существует множество полезных рабочих процессов, связанных с аксессорами. Например, сервис, который создает токены от имени другого сервиса, может хранить аксессор, связанный с конкретным идентификатором задания. Когда задание завершено, аксессор может быть использован для немедленного отзыва токена, выданного заданию, и всех его выданных учетных данных, что снижает вероятность того, что злоумышленник обнаружит и использует их.

1.5. Время жизни токена, периодические токены и явные максимальные TTL

Каждый не корневой токен имеет связанное с ним время жизни (TTL), которое представляет собой текущий период действия с момента создания или последнего обновления токена, в зависимости от того, что наступило позже. (Корневые токены могут иметь TTL, но их TTL также может быть равен 0, что означает, что срок действия токена никогда не истекает). После истечения текущего TTL токен больше не будет функционировать - он и связанные с ним аренды будут аннулированы.

Если токен является возобновляемым, можно продлить период действия токена, используя команду `starvault token renew` или соответствующую конечную точку продления. На результат выполнения данной операции влияют различные факторы. Что произойдет, зависит от того, является ли токен периодическим токеном (который могут создавать пользователи с правами *root*/*sudo*, роли хранилища токенов или некоторые методы аутентификации), имеет ли он явно указанный максимальный TTL или нет.

Периодические токены (Periodic tokens) — это токены, которые не имеют предустановленного максимального TTL и могут быть продлены неопределенное количество раз, при условии, что каждое продление не превышает интервал TTL, установленный для токена.

Токены с явно указанным максимальным TTL (Explicit maximum TTL) имеют установленный верхний предел времени жизни, который не может быть превышен, даже

если токен продлевается.

Если токен не является периодическим и не имеет явно указанного максимального TTL, то он может быть продлен до момента, когда суммарное время его жизни достигнет глобального максимального TTL, установленного для данного типа токенов в StarVault.

Общий случай

В общем случае, когда для токена не задан ни период, ни явное максимальное значение TTL, время жизни токена с момента его создания будет сравниваться с максимальным TTL. Это максимальное значение TTL генерируется динамически и может меняться от обновления к обновлению, поэтому значение не может быть отображено при поиске информации о токене. Оно основано на комбинации факторов:

1. Максимальный системный TTL, который составляет 32 дня, но может быть изменен в конфигурационном файле StarVault.
2. Максимальный TTL, установленный для точки монтирования. Это значение может перекрывать системный максимальный TTL - оно может быть длиннее или короче, и если оно установлено, то будет соблюдаться.
3. Значение, предложенное методом аутентификации, который выдал токен. Оно может быть настроено для каждой роли, группы или пользователя. Это значение может быть меньше максимального TTL точки монтирования (или, если оно не задано, максимального системного TTL), но не может быть больше.

Обратите внимание, что значения в п.2 и 3 могут измениться в любой момент времени, поэтому окончательное определение текущего разрешенного максимального TTL производится во время обновления с использованием текущих значений. Поэтому важно убедиться, что TTL, возвращаемый в результате операции обновления, находится в допустимом диапазоне. Если это значение не расширяется, то, скорее всего, TTL токена не может быть расширен за пределы его текущего значения, и клиент необходимо пройти повторную аутентификацию и получить новый токен.

Явные максимальные TTL

Для токенов может быть явно задан максимальный TTL. Это значение становится жестким ограничением на время жизни токена - независимо от того, каковы значения в п. 1, 2 и 3 из общего случая, токен не может жить дольше этого явно заданного значения. Это сказывается даже при использовании периодических токенов, чтобы избежать обычного механизма TTL.

Периодические токены

В некоторых случаях отзыв токена может быть проблематичным — например, если долгосрочной службе необходимо поддерживать пул подключений к SQL на протяжении длительного времени. В таком сценарии может быть использован периодический токен. Периодические токены могут быть созданы несколькими способами:

1. С использованием возможностей sudo или корневого токена в конечной точке **auth/token/create**.
2. С использованием роли хранилища токенов.
3. С использованием метода аутентификации, который поддерживает выдачу таких токенов, например **AppRole**.

В момент выдачи TTL периодического токена будет равен настроенному периоду. При каждом продлении TTL будет сбрасываться обратно к этому настроенному периоду, и если токен успешно продлевается в течение каждого из этих периодов времени, он никогда не истечет. За исключением корневых токенов, это в настоящее время единственный способ для токена в StarVault иметь неограниченный срок жизни.

1.6. Токены с привязкой к CIDR

Некоторые токены могут быть привязаны к CIDR, что ограничивает диапазон клиентских IP-адресов, которым разрешено их использовать. Это касается всех токенов, кроме неистекающих корневых токенов (тех, у которых TTL равен нулю). Если срок действия корневого токена истек, на него также распространяется действие CIDR-привязки.

2. Аутентификация с помощью метода Token

Этот метод аутентификации можно использовать для аутентификации через UI, CLI или API.

UI

На странице входа выберите **Token** в качестве метода и в поле **Token** введите соответствующий токен.

CLI

Используйте команду `starvault login`:

```
starvault login token=<token> ①
```

① <token> - соответствующий токен

3. Настройка встроенного метода аутентификации Token

Работа с токенами встроенного метода аутентификации возможна только с помощью CLI или API.

Для использования StarVault CLI подключитесь к любому серверу в кластере по SSH и аутентифицируйтесь с вашим токеном:

```
starvault login
Token (will be hidden):

Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "starvault login"
again. Future StarVault requests will automatically use this token.

<Output omitted>
```

При использовании самоподписанного сертификата ЦС его обязательно необходимо добавить в доверенные. Подробнее см. в разделе Подготовка рабочей среды.

В следующей таблице представлены доступные операции с токенами:

Команда	**Описание**
`starvault token –help`	Выводит справку по команде `starvault token`
`starvault token capabilities <token> <path>`	Используется для определения возможностей токена (`<token>`) на определенном пути (`<path>`) в StarVault. Возможности токена определяют, какие действия разрешены для пользователя или процесса, который аутентифицирован с использованием этого токена. Команда возвращает список возможностей, таких как чтение (`read`), запись (`write`), создание (`create`), удаление (`delete`) и обновление (`update`), которые токен имеет для данного пути.

Команда	Описание
`starvault token create [options]`	Создает новый токен, который может использоваться для аутентификации. Команда поддерживает следующие опции для настройки параметров токена: • `-display-name=<string>` - устанавливает отображаемое имя токена, которое будет показано в журналах аудита. • `-entity-alias=<string>` - используется для связывания создаваемого токена с существующим псевдонимом сущности (entity alias). Сущности представляют собой способ группировки одного или нескольких пользователей (или их учетных записей) в единую логическую идентификацию. Может использоваться только в комбинации с параметром `-role`. Кроме того, используемый псевдоним сущности должен быть перечислен в `allowed_entity_aliases` при настройке роли. • `-explicit-max-ttl=<duration>` - устанавливает максимальное время жизни (TTL), после которого токен не может быть продлен. • `-metadata=<key=value>` - добавляет метаданные к токену в формате ключ-значение. • `-no-default-policy` - указывает, что к токену не должна быть привязана политика по умолчанию. • `-orphan` - создает токен-сироту, который не будет иметь родительского токена. • `-period=<duration>` - устанавливает интервал, после которого токен может быть автоматически продлен. • `-policy=<string>` - привязывает одну или несколько политик к токену. Для указания нескольких политик достаточно перечислить их через запятую. • `-renewable` - указывает, может ли токен быть продлен. • `-role=<string>` - создает токен на основе указанной существующей роли.  Указание `-role` может переопределить другие аргументы команды. • `-ttl=<duration>` - устанавливает время жизни (TTL) токена. После истечения этого времени токен становится недействительным, если его не продлить. • `-type=<string>` - устанавливает тип токена (`service` или `batch`). • `-use-limit=<int>` - устанавливает ограничение на количество использований для создаваемого токена. Это число определяет, сколько раз токен может быть использован для аутентификации, прежде чем он станет недействительным.

Команда	Описание
`starvault token lookup [-accessor] [<token>]`	Получение информации об указанном токене. Если токен не указан, выводится информация о локальном токене. Команда поддерживает следующие опции: • `-accessor` - информация о токене будет получена через его аксессор.
`starvault token renew [-accessor] [-increment=<duration>] [<token>]`	Продлевает аренду указанного токена. Если токен не указан, продлевается аренда локального токена. Команда поддерживает следующие опции: • `-increment=<duration>` - позволяет указать на какой период необходимо продлить аренду. Значение игнорируется для периодических токенов. Если опция не указана, будет использоваться значение `default lease ttl`. • `-accessor` - операция будет выполняться через аксессор токена. При этом в выводе не будет включен токен.

`starvault token revoke [-accessor] [-mode=<string>] [-self] <token>`	Используется для отзыва указанного токена или всех токенов, связанных с определенной сущностью или ролью.
	
	Команда поддерживает следующие опции:
	
	• `-accessor` - операция будет выполняться через аксессор токена.
	• `-mode=<string>` - указывает режим отзыва. Если опция не указана, StarVault аннулирует токен и всех его детей. Допустимые значения:
	  ◦ `path` - отзывает все токены, связанные с определенным путем.
	  ◦ `orphan` - отзывает указанный токен, оставляя дочерние токены сиротами.
	• `-self` - отзыв текущего токена.

В примерах ниже представлены различные операции с токенами во встроенном методе аутентификации **Token**.

Пример 1. Создание простого токена без указания опций

```
starvault token create

Key                  Value
---                  -----
token                hvs.ypytfZ8LhbZEHsQFMk2LOSZF ①
token_accessor       VRDCwqcpxIB8urIRW3JXIb0N ②
token_duration       ∞ ③
token_renewable      false ④
```

```
token_policies       ["root"] ⑤
identity_policies    []
```

① Значение токена

② Значение аксессора токена

③ Срок действия токена. В данном примере он не ограничен

④ Возможность продления срока действия. В данном примере продление невозможно

⑤ Список политик, которые прикреплены к токену. В данном примере к токену прикреплена политика `root`

Пример 2. Просмотр информации о токене по его значению

```
starvault token lookup hvs.ypytfZ8LhbZEHsQFMk2LOSZF
Key                 Value
---                 -----
accessor            VRDCwqcpxIB8urIRW3JXIb0N
creation_time       1716889037
creation_ttl        0s
display_name        token
entity_id           n/a
expire_time         <nil>
explicit_max_ttl    0s
id                  hvs.ypytfZ8LhbZEHsQFMk2LOSZF
issue_time          2024-05-28T05:37:17.560963842-04:00
meta                <nil>
num_uses            0
orphan              false
path                auth/token/create
policies            [root]
renewable           false
ttl                 0s
type                service
```

Пример 3. Просмотр информации о токене по его аксессору

```
starvault token lookup -accessor VRDCwqcpxIB8urIRW3JXIb0N
Key                 Value
---                 -----
accessor            VRDCwqcpxIB8urIRW3JXIb0N
creation_time       1716889037
creation_ttl        0s
display_name        token
entity_id           n/a
expire_time         <nil>
explicit_max_ttl    0s
id                  n/a
```

```
issue_time          2024-05-28T05:37:17.560963842-04:00
meta                <nil>
num_uses            0
orphan              false
path                auth/token/create
policies            [root]
renewable           false
ttl                 0s
type                service
```

Пример 4. Создание токена с настройкой дополнительных параметров

В этом примере будет создан токен с ограничением на количество аутентификаций, а также с ограниченным TTL и явно заданным максимальным TTL.

```
starvault token create \
-ttl=5m \ ①
-explicit-max-ttl=15m \ ②
-use-limit=2 ③
```

① Срок действия токена установлен равным 5 минут

② Максимальный срок действия с учетом продления - 15 минут

③ С этим токеном можно аутентифицироваться 2 раза, после чего он будет аннулирован

Для проверки можно запросить информацию о токене:

```
starvault token lookup hvs.8MptwLC7AMesvipvAHlEPP0i

Key                 Value
---                 -----
accessor            aOAK1z4t4XR2QIWvK7UBOKmi
creation_time       1716898107
creation_ttl        5m
display_name        token
entity_id           n/a
expire_time         2024-05-28T08:13:27.725827029-04:00
explicit_max_ttl    15m ①
id                  hvs.8MptwLC7AMesvipvAHlEPP0i
issue_time          2024-05-28T08:08:27.725832484-04:00
meta                <nil>
num_uses            2 ②
orphan              false
path                auth/token/create
policies            [root]
renewable           true
ttl                 1m13s ③
type                service
```

① Максимальный срок действия токена

② Оставшееся количество попыток аутентификации

③ Оставшийся срок действия токена. Когда это значение достигнет нуля - токен аннулируется. Его продление после этого будет невозможно.

Пример 5. Продление аренды токена

В этом примере срок аренды токена будет продлен на 2 минуты:

```
starvault token renew -increment=2m hvs.rCsCab7MxAh0Ynmy6ZSWbqg8
```

С помощью просмотра информации о токене, можно убедиться, что для него обновились временные метки:

```
starvault token lookup hvs.rCsCab7MxAh0Ynmy6ZSWbqg8

Key                  Value
---                  -----
accessor             GL4i3xOx2MZq4fRbCmF1Bdac
creation_time        1716898659
creation_ttl         5m
display_name         token
entity_id            n/a
expire_time          2024-05-28T08:21:25.983575221-04:00 ①
explicit_max_ttl     15m
id                   hvs.rCsCab7MxAh0Ynmy6ZSWbqg8
issue_time           2024-05-28T08:17:39.248350428-04:00 ②
last_renewal         2024-05-28T08:19:25.983575317-04:00 ③
last_renewal_time    1716898765
meta                 <nil>
num_uses             2
orphan               false
path                 auth/token/create
policies             [root]
renewable            true
ttl                  1m57s
type                 service
```

① Дата и время истечения аренды токена

② Дата и время выдачи аренды

③ Дата и время последнего продления аренды

Пример 6. Создание токена, связанного с ролью

1. Создайте роль:

```
starvault write auth/token/roles/orphan orphan=true period=8h
```

2. Создайте токен на основе роли:

```
starvault token create -role=orphan
```

3. Проверьте конфигурацию токена:

```
starvault token lookup hvs.H64tiFk5dXc2tgP6ngDoiRPO

Key                 Value
---                 -----
accessor            KTwUuB73JEAK2M5Qk0Gm8rXp
creation_time       1716900987
creation_ttl        8h ①
display_name        token
entity_id           n/a
expire_time         2024-05-28T16:56:27.522692565-04:00
explicit_max_ttl    0s
id                  hvs.H64tiFk5dXc2tgP6ngDoiRPO
issue_time          2024-05-28T08:56:27.522697172-04:00
meta                <nil>
num_uses            0
orphan              true ①
path                auth/token/create/orphan
policies            [root]
renewable           true
role                orphan
ttl                 7h58m17s
type                service
```

① Параметры, полученные из роли