Механизм секретов Cubbyhole

Механизм секретов **cubbyhole** используется для хранения в настроенном физическом хранилище StarVault произвольных секретов, привязанных к токену. В **cubbyhole** пути ограничены областью действия каждого токена. Ни один токен не может получить доступ к **cubbyhole** другого токена. Когда токен истекает, его **cubbyhole** уничтожается.

В отличие от механизма **kv**, поскольку срок жизни **cubbyhole** связан со сроком жизни аутентификационного токена, нет понятия TTL (время жизни) или интервала обновления для значений, содержащихся в **cubbyhole** токена. Запись по ключу в механизме **cubbyhole** полностью заменяет старое значение.

1. Активация

Большинство механизмов должны быть предварительно сконфигурированы, прежде чем они смогут выполнять свои функции. Эти шаги обычно выполняются администратором или инструментом управления конфигурацией.

Для механизма секретов **cubbyhole** недоступно управление жизненным циклом. Он включен по умолчанию, его нельзя отключить, переместить или включить несколько раз.

2. Использование

После того как механизм секретов настроен и у пользователя/машины есть токен StarVault с соответствующими правами, он может генерировать учетные данные. Механизм секретов **cubbyhole** позволяет записывать ключи с произвольными значениями.

▶ UI

▶ CLI

Базы данных

Механизм секретов баз данных генерирует учетные данные динамически на основе настроенных ролей. Он работает с различными базами данных через интерфейс плагинов. Есть несколько встроенных типов баз данных, а также фреймворк для запуска пользовательских типов баз данных для расширения возможностей. Это означает, что сервисам, которым нужен доступ к базе данных, больше не нужно жестко кодировать учетные данные: они могут запрашивать их у StarVault и использовать механизм аренды StarVault для более удобного распространения ключей. Такие ключи называются "динамическими ролями" или "динамическими секретами".

Поскольку каждая служба обращается к базе данных с уникальными учетными данными, это значительно упрощает аудит при обнаружении сомнительного доступа к данным. Вы можете отследить его до конкретного экземпляра службы, основываясь на имени пользователя SQL.

StarVault использует свою собственную внутреннюю систему отзыва прав, чтобы гарантировать, что пользователи станут недействительными в течение разумного времени после истечения срока аренды.

1. Статические роли

При использовании динамических секретов StarVault генерирует уникальную пару имени пользователя и пароля для каждого уникального запроса учетных данных. StarVault также поддерживает статические роли для некоторых механизмов секретов баз данных. Статические роли - это сопоставление 1 к 1 ролей StarVault с именами пользователей в базе данных. При использовании статических ролей StarVault хранит и автоматически меняет пароли для связанного с ними пользователя базы данных в течение настраиваемого периода времени.

Когда клиент запрашивает учетные данные для статической роли, StarVault возвращает текущий пароль для того пользователя базы данных, который сопоставлен с запрашиваемой ролью. Благодаря статическим ролям любой человек с соответствующими политиками StarVault может получить доступ к связанной учетной записи пользователя в базе данных.

Не используйте статические роли для учетных данных root базы данных.

Не используйте те же учетные данные root базы данных, которые вы предоставляете StarVault in-config/ со статическими ролями. StarVault не делает различий между стандартными и root учетными данными при ротации паролей. Если вы назначите root учетные данные статической роли, все динамические или статические пользователи, управляемые этой конфигурацией базы данных, после ротации не будут работать, поскольку пароль для config/ больше не действителен.

Если вам нужно ротировать root учетные данные, используйте конечную точку API Rotate root credentials.

Обратитесь к таблице возможностей базы данных, чтобы определить, поддерживает ли выбранный вами бэкэнд базы данных статические роли.

2. Настройка

Большинство механизмов секретов должны быть предварительно сконфигурированы, прежде чем они смогут выполнять свои функции. Эти шаги обычно выполняются оператором или инструментом управления конфигурацией.

1. Включите механизм секретов базы данных:

   ```
   $ starvault secrets enable database
   Success! Enabled the database secrets engine at: database/
   ```

   По умолчанию механизм секретов будут включаться по имени механизма. Чтобы включить механизм секретов по другому пути, используйте аргумент `-path`.

2. Настройте StarVault с помощью соответствующего плагина и информации о подключении:

   ```
   $ starvault write database/config/my-database \
       plugin_name="..." \
       connection_url="..." \
       allowed_roles="..." \
       username="..." \
       password="..." \
   ```

   Настоятельно рекомендуется создать пользователя в базе данных специально для StarVault. Этот пользователь будет использоваться для работы с динамическими и статическими пользователями в базе данных. В документации этот пользователь называется `root`.

StarVault будет использовать указанного здесь пользователя для создания/обновления/отмены учетных данных базы данных. Этот пользователь должен обладать соответствующими правами на выполнение действий над другими пользователями базы данных (создание, обновление учетных данных, удаление и т. д.).

Этот механизм секретов может настраивать несколько подключений к базе данных. Для получения подробной информации о конкретных параметрах конфигурации обратитесь к документации по конкретной базе данных.

3. После настройки пользователя `root` настоятельно рекомендуется изменить пароль этого пользователя таким образом, чтобы пользователь starvault не был доступен никаким пользователям, кроме самого StarVault:

```bash
$ starvault write -force database/rotate-root/my-database
```

> После этого пароль для пользователя, указанного в предыдущем шаге, будет недоступен. В связи с этим настоятельно рекомендуется создать пользователя специально для StarVault, чтобы управлять пользователями базы данных.

4. Настройте роль, которая сопоставляет имя в StarVault с набором операторов создания для создания учетной записи базы данных:

```bash
$ starvault write database/roles/my-role \
    db_name=my-database \
    creation_statements="..." \
    default_ttl="1h" \
    max_ttl="24h"
Success! Data written to: database/roles/my-role
```

Поля `{{username}}` и `{{password}}` будут заполнены плагином динамически генерируемыми значениями. В некоторых плагинах также поддерживается поле `{{expiration}}`.

---

3. Использование

После того как механизм секретов настроен и у пользователя/машины есть токен StarVault с соответствующими правами, он может генерировать учетные данные.

Сгенерируйте новые учетные данные, считав из конечной точки `/creds` имя роли:

```bash
$ starvault read database/creds/my-role
```

Key	Value
`lease_id`	`database/creds/my-role/2f6a614c-4aa2-7b19-24b9-ad944a8d4de6`
`lease_duration`	`1h`
`lease_renewable`	`true`
`password`	`FSREZ1S0kFsZtLat-y94`
`username`	`v-vaultuser-e2978cd0-ugp7iqI2hdlff5hfjylJ-1602537260`

4. Возможности базы данных

В StarVault все базы данных поддерживают динамические и статические роли.

База данных	Ротация учетных данных `root`	Динамические роли	Статические роли	Настройка имени пользователя	Типы учетных данных
Cassandra	Да	Да	Да	Да	пароль
InfluxDB	Да	Да	Да	Да	пароль
MSSQL	Да	Да	Да	Да	пароль
MySQL/MariaDB	Да	Да	Да	Да	пароль
Oracle	Да	Да	Да	Да	пароль
PostgreSQL	Да	Да	Да	Да	пароль

5. Кастомные плагины

Этот механизм секретов позволяет запускать пользовательские типы баз данных через открытый интерфейс плагина. Более подробную информацию вы можете найти в разделе "Плагин для пользовательских баз данных".

6. Типы учетных данных

Системы баз данных поддерживают различные методы аутентификации и типы учетных данных. Механизм секретов базы данных поддерживает управление учетными данными, альтернативными именам пользователей и паролям. Параметры credential_type и credential_-config динамических и статических ролей определяют тип учетных данных, которые StarVault будет генерировать и предоставлять плагинам баз данных. О том, какие типы учетных данных они поддерживают, и о примерах их использования читайте в документации к отдельным плагинам баз данных.

7. Генерация пароля

Пароли генерируются с помощью политик паролей. Базы данных могут опционально устанавливать политику паролей для использования во всех ролях или на уровне отдельных ролей для данной базы данных.

Например, при каждом вызове `starvault write database/config/my-database` вы можете задать политику паролей для всех ролей, использующих `my-database`. Каждая база данных имеет политику паролей по умолчанию, которая определяется следующим образом: 20 символов, из которых как минимум 1 символ в верхнем регистре, минимум 1 символ в нижнем регистре, минимум 1 число и минимум 1 символ тире.

Генерация паролей по умолчанию может быть представлена в виде следующей политики паролей:

```
length = 20

rule "charset" {
    charset = "abcdefghijklmnopqrstuvwxyz"
    min-chars = 1
}
rule "charset" {
    charset = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    min-chars = 1
}
rule "charset" {
    charset = "0123456789"
    min-chars = 1
}
rule "charset" {
    charset = "-"
    min-chars = 1
}
```

8. Отключение экранирования символов

Вы можете указать опцию `disable_escaping` со значением `true` в некоторых механизмах секретов, чтобы запретить StarVault экранировать специальные символы в полях имени пользователя и пароля. Это необходимо для некоторых альтернативных форматов строк подключения, таких как ADO с MSSQL. Ознакомьтесь с к документациям по API движку секретов баз данных и по отдельным плагинам, чтобы определить поддержку этого параметра.

Например, если пароль содержит символы URL-escaped, такие как `#` или `%`, они останутся такими, а не превратятся в `%23` и `%25` соответственно.

BASH

```
$ starvault write database/config/my-mssql-database \
plugin_name="mssql-database-plugin" \
connection_url='server=localhost;port=1433;user id={{username}};password=
{{password}};database=mydb;' \
username="root" \
password='your#StrongPassword%' \
disable_escaping="true"
```

9. API

Движок секретов базы данных имеет полный HTTP API. Более подробную информацию см. в разделе API механизма секретов баз данных.

Механизм секретов Identity

Механизм секретов Identity — это решение для управления идентификацией в StarVault. Внутри него хранятся клиенты, которые распознаются StarVault. Каждый клиент называется сущностью. Сущность может иметь несколько псевдонимов. Например, один пользователь, имеющий учетные записи на userpass и LDAP, может быть сопоставлен с одной сущностью в StarVault, которая имеет 2 псевдонима: один типа userpass и один типа LDAP. Когда клиент проходит аутентификацию через любой из бэкендов (кроме бэкенда Token), StarVault создает новую сущность и прикрепляет к ней новый псевдоним, если соответствующая сущность еще не существует. Идентификатор сущности будет привязан к аутентифицированному токену. Когда такие токены используются, их идентификаторы сущностей регистрируются в журнале аудита, что позволяет отследить действия, совершенные конкретными пользователями.

Хранилище идентификаторов позволяет операторам управлять сущностями в StarVault. Можно создавать сущности и привязывать к ним псевдонимы с помощью ACL'd API. Для сущностей могут быть установлены политики, которые добавляют возможности токенам, привязанным к идентификаторам сущностей. Возможности, предоставляемые токенам через сущности, являются дополнением к существующим возможностям токена, а не заменой. Возможности токена, наследуемые от сущностей, вычисляются динамически во время запроса. Это обеспечивает гибкость в управлении доступом к уже выпущенным токенам.

Этот механизм секретов будет установлен по умолчанию. Его нельзя отключить или переместить. Для получения более подробной информации об идентификации обратитесь к документации Identity.

Механизм секретов StarVault Identity поддерживает несколько различных функций. Каждая функция описана на отдельной странице документации:

- токены Identity
- OIDC Identity провайдер

1. Удостоверяющие токены

1.1. Введение

Сведения об удостоверениях используется во всем StarVault, но ее также можно экспортировать для использования другими приложениями. Авторизованный пользователь

или приложение может запросить токен, содержащий удостоверяющую информацию для связанного с ним объекта. Эти токены подписаны JWT, соответствуя структуре токена OIDC ID. Ключи, используемые для аутентификации токенов, публикуются StarVault на неаутентифицированной конечной точке в соответствии с OIDC discovery и соглашениями JWKS, которые должны быть непосредственно использованы библиотеками JWT/OIDC. StarVault также предоставляет конечную точку для проверки токенов.

1.2. Роли и ключи

OIDC-совместимые ID-токены генерируются на основе роли, которая позволяет настроить токену с помощью системы шаблонов, ttl токена и указать, какой «ключ» будет использоваться для подписи токена. Шаблон роли является необязательным параметром для настройки содержимого токена и описывается в следующем разделе. TTL токена контролирует время действия токена, по истечении которого библиотеки верификации будут считать токен недействительным. Все роли имеют связанный с ними `client_id`, который будет добавлен к параметру токена `aud`. Библиотекам JWT/OIDC обычно требуется это значение. Параметр может быть установлен оператором в выбранное значение, либо будет использовано значение, сгенерированное StarVault, если оно не задано.

Параметр `key` роли связывает роль с существующим именованным ключом (несколько ролей могут ссылаться на один и тот же ключ). Невозможно сгенерировать неподписанный ID-токен.

Именованный ключ - это пара открытый/закрытый ключ, сгенерированная StarVault. Закрытый ключ используется для подписи удостоверяющих токенов, а открытый ключ используется клиентами для проверки подписи. Ключи регулярно ротируются, при этом генерируется новая пара ключей, а предыдущий открытый ключ сохраняется в течение ограниченного времени.

В конфигурации именованного ключа указываются период ротации, время проверки, алгоритм подписи и разрешенные удостоверения клиентов. Период ротации - это частота, с которой генерируется новый ключ подписи и удаляется закрытая часть предыдущего ключа подписи. Проверка ttl - это время, в течение которого открытый ключ сохраняется для проверки после ротации. По умолчанию ключи ротируются каждые 24 часа и остаются доступными для проверки в течение 24 часов после ротации.

Список разрешенных удостоверений клиентов ключа ограничивает, какие роли могут ссылаться на этот ключ. Параметр может быть установлен в `*`, чтобы разрешить все роли. Оценка действительности производится при запросе токена, а не во время настройки.

1.3. Содержание и шаблоны токенов

Удостоверяющие токены всегда будут содержать, как минимум, запросы OIDC:

- `iss` - URL-адрес издателя

- `sub` - идентификатор объекта организации-заявителя
- `aud` - `client_id` для роли
- `iat` - время выпуска
- `exp` - время истечения срока действия токена

Кроме того, оператор может настроить шаблоны для каждой роли, которые позволяют добавлять в токен другую информацию об объектах. Шаблоны структурированы в виде JSON с заменяемыми параметрами. Синтаксис параметров такой же, как и в шаблонах ACL Path Templating.

Например:

BASH

```
{
  "color": {{identity.entity.metadata.color}},
  "userinfo": {
     "username": {{identity.entity.aliases.usermap_123.metadata.username}},
     "groups": {{identity.entity.groups.names}}
  },
  "nbf": {{time.now}}
}
```

Когда запрашивается токен, результирующий шаблон может быть заполнен следующим образом:

BASH

```
{
  "color": "green",
  "userinfo": {
     "username": "bob",
     "groups": ["web", "engr", "default"]
  },
  "nbf": 1561411915
}
```

которые будут объединены с базовыми запросами OIDC в окончательный токен:

BASH

```
{
  "iss": "https://10.1.1.45:8200/v1/identity/oidc",
  "sub": "a2cd63d3-5364-406f-980e-8d71bb0692f5",
  "aud": "SxSouteCYPBoaTFy94hFghmekos",
  "iat": 1561411915,
  "exp": 1561412215,
  "color": "green",
  "userinfo": {
    "username": "bob",
    "groups": ["web", "engr", "default"]
  },
```

```
  "nbf": 1561411915
}
```

Обратите внимание, как происходит слияние шаблонов, при котором ключи верхнего уровня шаблона становятся ключами верхнего уровня токена. По этой причине шаблоны не могут содержать ключи верхнего уровня, которые перезаписывают стандартные OIDC запросы.

Параметры шаблона, которые отсутствуют в объекте, например, метаданные или аксессор псевдонима, являются просто пустыми строками или объектами, в зависимости от типа данных.

Шаблоны настраиваются на роль и могут быть дополнительно закодированы в base64.

Полный список параметров шаблона приведен ниже:

Name	**Description**
identity.entity.id	Удостоверение организации
identity.entity.name	Имя объекта
identity.entity.groups.ids	Удостоверение групп, в которых состоит объект
identity.entity.groups.names	Имена групп, в которых состоит объект
identity.entity.metadata	Метаданные, связанные с объектом
identity.entity.metadata.<metadata key>	Метаданные, связанные с объектом для данного ключа
identity.entity.aliases.<mount accessor>.id	Удостоверение псевдонима объекта для данной точки монтирования
identity.entity.aliases.<mount accessor>.name	Имя псевдонима объекта для данной точки монтирования
identity.entity.aliases.<mount accessor>.metadata	Метаданные, связанные с псевдонимом для данной точки монтирования
identity.entity.aliases.<mount accessor>.metadata.<metadata key>	Метаданные, связанные с псевдонимом для данной точки монтирования и ключа метаданных
identity.entity.aliases.<mount accessor>.custom_metadata	Пользовательские метаданные, связанные с псевдонимом для данной точки монтирования
identity.entity.aliases.<mount accessor>.custom_metadata.<custom_metadata key>	Пользовательские метаданные, связанные с псевдонимом для данной точки монтирования и ключом пользовательских метаданных
time.now	Текущее время в виде секунд с начала эпохи

time.now.plus.<duration>	Текущее время плюс продолжительность в формате строки
time.now.minus.<duration>	Текущее время минус продолжительность в формате строки

1.4. Генерация токенов

Аутентифицированный клиент может запросить токен, используя конечную точку генерации токена. Токен будет сгенерирован в соответствии со спецификациями запрашиваемой роли для запрашивающего объекта. Невозможно сгенерировать токен для произвольного объекта.

1.5. Проверка подлинности удостоверяющих токенов, сгенерированных StarVault

Удостоверяющий токен может быть проверен клиентской стороной с помощью открытых ключей, опубликованных StarVault, или через конечную точку самопроверки, предоставленную StarVault.

StarVault предоставляет стандартные **.well-known** конечные точки, которые позволяют легко интегрировать их с библиотеками проверки OIDC. Настройка библиотек обычно включает в себя предоставление URL-адреса издателя и удостоверения клиента. Затем библиотека будет обрабатывать запросы ключей и проверять подписи и требования к запросам на токенах. Преимущество этого подхода заключается в том, что требуется только доступ к StarVault, а не авторизация, поскольку **.well-known** конечные точки не проходят аутентификацию.

В качестве альтернативы токен может быть отправлен в StarVault для проверки через конечную точку самопроверки. В ответе будет указано, является ли токен «активным» или нет, а также все ошибки, возникшие при проверке. Помимо того, что клиент может просто передать проверку StarVault, использование этой конечной точки включает дополнительную проверку того, активен ли объект до сих пор, что невозможно определить только по токену. В отличие от **.well-known** конечной точки, для доступа к конечной точке самопроверки требуется действительный токен StarVault и достаточная авторизация.

1.6. Соображения по поводу издателя

Система токенов удостоверения имеет один настраиваемый параметр: издателя. Запрос идентификатора издателя токена `iss` особенно важно для правильной проверки токена клиентами, и ему следует уделить особое внимание при удостоверении токенов с репликацией производительности. Потребители токена запрашивают открытые ключи у StarVault, используя URL-адрес издателя, поэтому он должен быть доступен по сети. Кроме

того, возвращаемый набор ключей будет включать издателя, который должен соответствовать запросу.

По умолчанию StarVault устанавливает издателя на api_addr экземпляра StarVault. Это означает, что токены, выпущенные в данном кластере, должны быть проверены в этом же кластере. В качестве альтернативы параметр issuer может быть настроен явным образом. Этот адрес должен вести к конечной точке `identity/oidc` данного экземпляра StarVault (например, https://starvault-1.example.com:8200/v1/identity/oidc) и должен быть доступен любому клиенту, пытающемуся подтвердить идентификационные токены.

2. Идентифицирующий провайдер OIDC

StarVault является провайдером идентификации OpenID Connect (OIDC). Это позволяет клиентским приложениям, использующим протокол OIDC, использовать источник идентификации StarVault и широкий спектр методов аутентификации при проверке подлинности конечных пользователей. Клиентские приложения могут настроить свою логику аутентификации на взаимодействие с StarVault. После этого StarVault будет выступать в качестве моста к другим поставщикам идентификационных данных с помощью существующих методов аутентификации. Клиентские приложения также могут получать идентификационную информацию для своих конечных пользователей, используя пользовательский шаблон идентификационной информации StarVault.

Более подробную информацию о ресурсах конфигурации и конечных точках OIDC можно найти на странице концепций провайдеров OIDC.

2.1. Настройка

Система провайдеров StarVault OIDC построена на основе механизма секретов идентификации. Этот механизм секретов установлен по умолчанию, его нельзя отключить или переместить.

Каждое пространство имен StarVault имеет провайдера и ключ OIDC по умолчанию. Эта встроенная конфигурация позволяет клиентским приложениям начать использовать StarVault в качестве источника идентификации с минимальными настройками. Подробные сведения о встроенной конфигурации и дополнительных опциях см. на странице Концепции провайдера OIDC.

В следующих шагах показана минимальная конфигурация, позволяющая клиентскому приложению использовать StarVault в качестве провайдера OIDC.

1. Включите метод авторизации StarVault:

```
$ starvault auth enable userpass
Success! Enabled userpass auth method at: userpass/
```

В потоке OIDC можно использовать любой метод авторизации StarVault. Для простоты включите метод аутентификации `userpass`.

2. Создайте пользователя

```
$ starvault write auth/userpass/users/end-user password="securepassword"
Success! Data written to: auth/userpass/users/end-user
```

Этот пользователь будет проходить аутентификацию в StarVault через клиентское приложение, иначе называемое доверенной стороной OIDC.

3. Создайте клиентское приложение:

```
$ starvault write identity/oidc/client/my-webapp \
  redirect_uris="https://localhost:9702/auth/oidc-callback" \
  assignments="allow_all"
Success! Data written to: identity/oidc/client/my-webapp
```

Эта операция создает клиентское приложение, которое можно использовать для настройки полагающейся стороны OIDC. Подробные сведения о различных типах клиентов, включая `confidential` и `public`, см. в разделе "Клиентские приложения".

Параметр `assignments` ограничивает сущности и группы StarVault, которым разрешена аутентификация через клиентское приложение. По умолчанию ни одной сущности StarVault не разрешено. Чтобы разрешить аутентификацию всем сущностям StarVault, используется встроенное назначение `allow_all`.

4. Прочтите учетные данные клиента

```
$ starvault read identity/oidc/client/my-webapp
```

Вывод:

Key	Value
access_token_ttl	24h
assignments	[allow_all]
client_id	GSDTnn3KaOrLpNlVGlYLS9TVsZgOTweO
client_secret	hvo_secret_gBKHcTP58C4aq7FqPWsuqKgpiiegd7ah-pifGae9WGkHRCwFEJTZA9KGdNVpzE0r8

client_type	confidential
id_token_ttl	24h
key	default
redirect_uris	[https://localhost:9702/auth/oidc-callback]

`Client_id` и `client_secret` - это учетные данные клиентского приложения. Эти значения обычно требуются при настройке доверяющей стороны OIDC.

5. Прочтите конфигурацию обнаружения OIDC:

```bash
$ curl -s http://127.0.0.1:8200/v1/identity/oidc/provider/default/.well-known/openid-configuration
{
  "issuer": "http://127.0.0.1:8200/v1/identity/oidc/provider/default",
  "jwks_uri": "http://127.0.0.1:8200/v1/identity/oidc/provider/default/.well-known/keys",
  "authorization_endpoint": "http://127.0.0.1:8200/ui/vault/identity/oidc/provider/default/authorize",
  "token_endpoint": "http://127.0.0.1:8200/v1/identity/oidc/provider/default/token",
  "userinfo_endpoint": "http://127.0.0.1:8200/v1/identity/oidc/provider/default/userinfo",
  "request_parameter_supported": false,
  "request_uri_parameter_supported": false,
  "id_token_signing_alg_values_supported": [
    "RS256",
    "RS384",
    "RS512",
    "ES256",
    "ES384",
    "ES512",
    "EdDSA"
  ],
  "response_types_supported": [
    "code"
  ],
  "scopes_supported": [
    "openid"
  ],
  "subject_types_supported": [
    "public"
  ],
  "grant_types_supported": [
    "authorization_code"
  ],
  "token_endpoint_auth_methods_supported": [
    "none",
```

```
    "client_secret_basic",
    "client_secret_post"
  ],
  "code_challenge_methods_supported": [
    "plain",
    "S256"
  ]
}
```

Каждый провайдер StarVault OIDC публикует метаданные обнаружения. Значение `issuer` обычно требуется при настройке полагающейся стороны OIDC.

2.2. Использование

После настройки метода аутентификации StarVault и клиентского приложения следующие сведения можно использовать для настройки доверяющей стороны OIDC на делегирование аутентификации конечного пользователя в StarVault.

- `cliend_id` - идентификатор клиентского приложения
- `client_secret` - секрет клиентского приложения
- `issuer` - эмитент (issuer) провайдера OIDC в StarVault

Обратитесь к документации конкретного OIDC-получателя (relying party) для получения сведений об использовании.

2.3. Поддерживаемые потоки

Функциональность OIDC-провайдера в StarVault в настоящее время поддерживает следующий поток аутентификации:

- поток авторизационного кода (Authorization Code Flow)

3. API

Движок секретов Identity имеет полноценный HTTP API. Более подробную информацию можно найти в разделе API механизма секретов Identity.

Кроме того, StarVault может быть настроен как провайдер идентификации OIDC. Более подробную информацию можно найти в API провайдера идентификации OIDC.