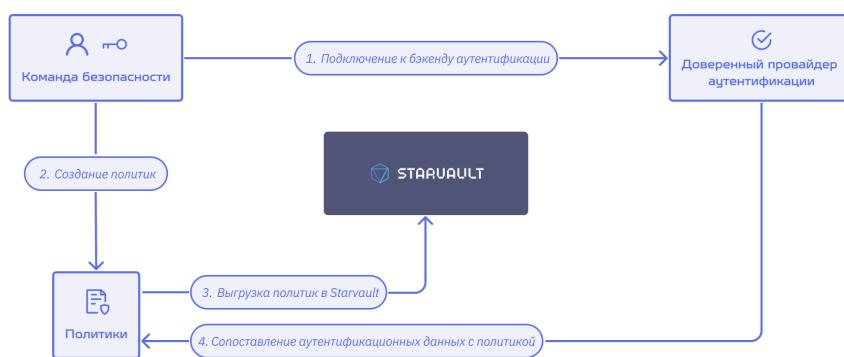


# Политики доступа

## 1. Процесс авторизации с политиками

Прежде чем человек или машина получат доступ, администратор должен настроить StarVault с использованием метода аутентификации. **Аутентификация** — это процесс, в ходе которого информация, предоставленная человеком или машиной, проверяется на соответствие внутренней или внешней системе.



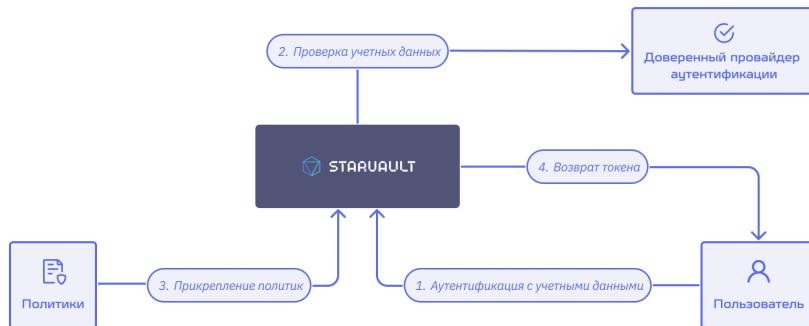
На диаграмме выше представлены шаги, которые необходимо выполнить для настройки StarVault:

1. Команда безопасности настраивает StarVault на подключение к методу аутентификации. Эта настройка зависит от метода аутентификации. Например, для LDAP StarVault должен знать адрес сервера LDAP и необходимость подключения по TLS.
2. Команда безопасности создает политику (или использует существующую политику), которая предоставляет доступ к путям в StarVault. Политики пишутся на языке HCL в выбранном вами редакторе и сохраняются на диске.
3. Содержимое политики загружается и хранится в StarVault, а ссылки на нее даются по имени. Имя политики можно рассматривать как указатель или символьическую ссылку на ее набор правил.
4. Самое главное, что команда безопасности сопоставляет данные в методе аутентификации с политикой. Например, команда безопасности может создать такие сопоставления, как:
  - Члены группы "dev" сопоставляются с политикой StarVault с именем "readonly-dev".
  - Члены группы "ops" сопоставляются с политиками StarVault "admin" и "auditor".

или

- Члены группы "ops" сопоставляются с политиками StarVault "admin" и "auditor".

После выполнения описанных операций в StarVault есть сопоставление между внутренней системой аутентификации и внутренней политикой. Когда пользователь аутентифицируется в StarVault, фактическая аутентификация делегируется методу аутентификации. Для пользователя этот процесс выглядит следующим образом:



1. Пользователь пытается аутентифицироваться в StarVault с помощью своих учетных данных, предоставляя StarVault свое имя пользователя и пароль, в соответствии с выбранным методом аутентификации (например, LDAP).
2. StarVault устанавливает соединение с соответствующим провайдером идентификации (например, LDAP) и просит сервер проверить предоставленные учетные данные. Если проверка прошла успешно, сервер возвращает информацию о пользователе, включая группы, к которым он относится.
3. StarVault сопоставляет результаты с сервера с политиками внутри StarVault, используя сопоставление, настроенное командой безопасности в предыдущем разделе. Затем StarVault генерирует токен и присоединяет соответствующие политики.
4. StarVault возвращает токен пользователю. Этому токену назначены правильные политики, как это предусмотрено конфигурацией сопоставления, которая была заранее настроена командой безопасности.

Затем пользователь использует этот токен StarVault для последующих операций. Если пользователь снова выполнит шаги аутентификации, он получит новый токен. У токена будут те же права, но сам токен будет другим. Повторная аутентификация не аннулирует исходный токен.

## 2. Синтаксис политик

Политики пишутся на языке HCL или JSON и описывают, к каким путям в StarVault разрешен доступ пользователю или машине.

Ниже представлен пример простой политики, которая предоставляет возможность чтения ( read ) для пути KV v1 "secret/foo":

```
path "secret/foo" {
    capabilities = ["read"]
}
```

Когда эта политика назначена токену, токен может читать данные из "secret/foo". Однако токен не может обновлять или удалять "secret/foo", поскольку политика не предоставляет таких возможностей. Политики по умолчанию запрещают доступ, поэтому токен не будет иметь другого доступа в StarVault.

Ниже представлен пример более детализированной политики:

```
# Этот раздел выдаёт права на любые действия в точке "secret/*".
# На этот доступ можно наложить дополнительные ограничения.
path "secret/*" {
    capabilities = ["create", "read", "update", "patch", "delete", "list"]
}

# Несмотря на полный доступ к "secret/*", выданный выше,
# этот раздел явно запрещает любые операции с "secret/super-secret".
# Это правило будет иметь более высокий приоритет
path "secret/super-secret" {
    capabilities = ["deny"]
}

# Политики также могут содержать разрешенные, запрещенные и обязательные
# параметры
# В этом примере разрешено создание секрета по пути "secret/restricted",
# но при этом секрет может содержать только:
# - ключ "foo" с любыми значениями
# - ключ "bar" со значением "zip" или "zap"
path "secret/restricted" {
    capabilities = ["create"]
    allowed_parameters = {
        "foo" = []
        "bar" = ["zip", "zap"]
    }
}
```

Политики используют сопоставление по пути для проверки набора возможностей по запросу. Путь в политике может указывать на точное соответствие пути или может использовать шаблоны glob-символами, который указывает StarVault на необходимость сопоставления по префикску:

```
# Разрешено чтение только "secret/foo".
# Прикрепленный токен не сможет читать "secret/food" или "secret/foo/bar".
path "secret/foo" {
    capabilities = ["read"]
```

```
}

# Разрешено чтение по любым путям в "secret/bar".
# Например, токен сможет читать данные в
# "secret/bar/zip" и "secret/bar/zip/zap", но не в "secret/bars/zip".
path "secret/bar/*" {
    capabilities = ["read"]
}

# Разрешено чтение по пути с префиксом "zip-".
# Например, токен сможет читать данные в
# "secret/zip-zap" или "secret/zip-zap/zong", но не в "secret/zip/zap"
path "secret/zip-*" {
    capabilities = ["read"]
}
```

Кроме того, знак + может использоваться для обозначения любого количества символов, ограниченного одним сегментом пути:

```
# Разрешено чтение по пути "teamb" в любом пути верхнего уровня в "secret/"
path "secret/+/teamb" {
    capabilities = ["read"]
}

# Разрешено чтение по пути "teamb", находящемся на 3 уровня ниже "secret/"
# Например, secret/foo/bar/teamb, secret/bar/foo/teamb, и т.д.
path "secret/+/+/teamb" {
    capabilities = ["read"]
}
```

Архитектура StarVault похожа на файловую систему. Каждое действие в StarVault имеет соответствующий путь и возможность - даже внутренние конечные точки конфигурации ядра StarVault находятся по пути "sys/". Политики определяют доступ к этим путям и возможностям, что контролирует доступ токена к данным в StarVault.

**!** Правила политики, которые применяет StarVault, определяются по наиболее точному из доступных совпадений, используя правила приоритета, описанные ниже. Это может быть точное совпадение или совпадение по самому длинному префиксу в шаблоне. Если один и тот же шаблон встречается в нескольких политиках, возможности объединяются. Если в соответствующих политиках встречаются разные шаблоны, используется только наиболее приоритетное совпадение из этих политик.

Это означает, что если вы определите политику для "secret/foo\*", она также будет соответствовать "secret/foobar". В частности, когда потенциально существует несколько совпадающих путей политики, P1 и P2, применяются следующие критерии совпадения:

1. Если первый знак `+` или `*` встречается раньше в P1, P1 имеет более низкий приоритет.
2. Если P1 заканчивается на `*`, а P2 - нет, P1 имеет более низкий приоритет.
3. Если P1 имеет больше сегментов `+`, P1 имеет более низкий приоритет.
4. Если P1 короче, он имеет более низкий приоритет.
5. Если P1 меньше лексикографически, то он имеет более низкий приоритет

Например, если взять два пути, `secret/*` и `secret///foo/*`, то первый знак `\`` появляется в одном и том же месте, оба оканчиваются на `\`*\``, а второй имеет два сегмента `\``, в то время как первый - ноль. Поэтому `secret///foo/*` будет иметь более низкий приоритет.

**!** Символ glob, упоминаемый в этой документации, - это звездочка (`*`). Он не является регулярным выражением и поддерживается только в качестве последнего символа пути!

**!** При предоставлении возможности работы с операцией `list` важно отметить, что поскольку `list` всегда оперирует префиксом, политики также должны оперировать префиксом, поскольку StarVault будет проверять пути запросов на наличие префиксов.

### 3. Возможности

Каждый путь должен определять одну или несколько возможностей, которые обеспечивают тонкий контроль над разрешенными (или запрещенными) операциями. Как показано в примерах выше, возможности всегда указываются в виде списка строк, даже если существует только одна возможность.

Чтобы определить возможности, необходимые для выполнения конкретной операции, к подкоманде CLI можно добавить флаг `--output-policy`. Например:

```
starvault auth enable --output-policy userpass

path "sys/auth/userpass" {
    capabilities = ["create", "update", "sudo"]
}
```

```
starvault auth disable --output-policy userpass

path "sys/auth/userpass" {
    capabilities = ["delete", "sudo"]
}
```

Список возможностей включает следующее:

- `create` (соответствует запросу POST / PUT для API) - Позволяет создавать данные по заданному пути. Очень немногие компоненты StarVault различают создание и обновление, поэтому для большинства операций требуются возможности как создания, так и обновления.
- `read` (соответствует запросу GET для API) - Позволяет считывать данные по заданному пути.
- `update` (соответствует запросу POST / PUT для API) - Позволяет изменять данные по заданному пути. В большинстве компонентов StarVault это неявно включает возможность создания начального значения в пути.
- `patch` (соответствует запросу PATCH для API) - Позволяет частично обновлять данные на заданном пути.
- `delete` (соответствует запросу DELETE для API) - Позволяет удалить данные по указанному пути.
- `list` (соответствует запросу LIST для API) - Позволяет просматривать список значений по заданному пути. Обратите внимание, что ключи, возвращаемые операцией `list`, не фильтруются политиками. Не кодируйте конфиденциальную информацию в именах ключей. Не все бэкенды поддерживают `list`.

В дополнение к стандартному набору есть некоторые специальные возможности (они не имеют сопоставления с запросами API):

- `sudo` - Позволяет получить доступ к путям, защищенным правами `root`. Токенам не разрешается взаимодействовать с этими путями, если они не обладают правами sudo (в дополнение к другим правам, необходимым для выполнения операции над этим путем, например, `read` или `delete`).

Например, для изменения бэкендов журнала аудита требуется токен с привилегиями sudo.

- `deny` - Запрещает доступ. Этот параметр всегда имеет приоритет, независимо от любых других определенных возможностей, включая `sudo`.

## 4. Шаблонизация политики

Синтаксис политик в StarVault позволяет выполнять замену переменных в некоторых строках политик на значения, доступные для токена. В настоящее время можно вводить идентификационную информацию в пути политик.

Для шаблонизации политик доступны следующие параметры:

Имя параметра	Описание
identity.entity.id	Идентификатор сущности
identity.entity.name	Имя сущности
identity.entity.metadata.<metadata key>	Метаданные, связанные с сущностью для заданного ключа
identity.entity.aliases.<mount accessor>.id	Идентификатор псевдонима сущности для заданной точки монтирования
identity.entity.aliases.<mount accessor>.name	Имя псевдонима сущности для заданной точки монтирования
identity.entity.aliases.<mount accessor>.metadata.<metadata key>	Метаданные, связанные с псевдонимом для заданной точки монтирования и ключа метаданных
identity.entity.aliases.<mount accessor>.custom_metadata.<custom_metadata key>	Пользовательские метаданные, связанные с псевдонимом для заданной точки монтирования и пользовательского ключа метаданных
identity.groups.ids.<group id>.name	Имя группы для заданного идентификатора группы
identity.groups.names.<group name>.id	Идентификатор группы для заданного имени группы
identity.groups.ids.<group id>.metadata.<metadata key>	Метаданные, связанные с идентификатором группы для заданного ключа
identity.groups.names.<group name>.metadata.<metadata key>	Метаданные, связанные с именем группы для данного ключа

## Примеры

**Пример 1.** Следующая политика создает раздел механизма KV v2 для определенного пользователя

```
path "secret/data/{{identity.entity.id}}/*" {
    capabilities = ["create", "update", "patch", "read", "delete"]
}

path "secret/metadata/{{identity.entity.id}}/*" {
    capabilities = ["list"]
}
```

## Пример 2. Создание общего раздела KV, связанного с сущностями, входящими в группу.

```
# В примере идентификатор группы сопоставляется с именем группы
path "secret/data/groups/{{identity.groups.ids.fb036ebc-2f62-4124-9503-
42aa7A869741.name}}/*" {
    capabilities = ["create", "update", "patch", "read", "delete"]
}

path "secret/metadata/groups/{{identity.groups.ids.fb036ebc-2f62-4124-
9503-42aa7A869741.name}}/*" {
    capabilities = ["list"]
}
```

Если вы хотите использовать метаданные, связанные с плагином аутентификации, в своих шаблонах, вам нужно получить его аксессор монтирования и получить к нему доступ через ключ `aliases`.

Значение аксессора монтирования можно получить с помощью следующей команды:

starvault auth list			
Path	Type	Accessor	Description
kubernetes/ token/	kubernetes token	auth_kubernetes_xxxx auth_token_yyyy	n/a token based credentials

Следующая шаблонизированная политика позволяет читать путь, связанный с пространством имен учетной записи сервиса Kubernetes для идентификатора:

```
path
"secret/data/{{identity.entity.aliases.auth_kubernetes_xxxx.metadata.service_ac-
count_namespace}}/*" {
    capabilities = ["read"]
}
```

## 5. Тонкий контроль

В дополнение к стандартному набору возможностей StarVault предлагает более тонкий контроль над разрешениями на определенном пути. Возможности, связанные с путем, имеют приоритет над разрешениями на параметры.



Поля `allowed_parameters`, `denied_parameters` и `required_parameters` не поддерживаются для политик, используемых с механизмом секретов kv версии 2.

Политики могут учитывать параметры запроса, чтобы ввести дополнительные ограничения, используя следующие опции:

- `required_parameters` - Список параметров, которые необходимо указать.

Например:

```
# Следующая запись разрешает пользователю создание секрета по пути  
"secret/profile".  
# При этом требуется, чтобы создаваемый секрет содержал ключи "name" и "id".  
path "secret/profile" {  
    capabilities = ["create"]  
    required_parameters = ["name", "id"]  
}
```

- `allowed_parameters` - Список ключей и значений, которые разрешены на данном пути. Возможны различные способы использования данного поля:

- Установка параметра со значением из пустого списка позволяет параметру содержать любое значение.

```
# Эта запись разрешает пользователю обновлять пароль любого пользователя  
# настроенного в userpass.  
# При этом значение пароля может быть любым.  
# Однако, этот пользователь не сможет изменять другие параметры, такие  
# как "token_ttl"  
path "auth/userpass/users/*" {  
    capabilities = ["update"]  
    allowed_parameters = {  
        "password" = []  
    }  
}
```

- Указание списка значений позволяет ограничить допустимые значения, которые могут быть присвоены параметру.

```
# В этом примере пользователь может создавать и обновлять транзитные  
# ключи.  
# Разрешено устанавливать только параметр "auto_rotate_period"  
# при этом он может принимать значения только "8h", "24h" или "5d"  
  
path "transit/keys/*" {  
    capabilities = ["create", "update"]  
    allowed_parameters = {  
        "auto_rotate_period" = ["8h", "24h", "5d"]  
    }  
}
```

- Если указаны какие-либо ключи, все не указанные параметры будут запрещены, за исключением случая, когда для параметра "\*" в качестве значения установлен пустой массив ( [] ), что позволит изменять все остальные параметры. Параметры с определенными значениями будут по-прежнему ограничены этими значениями.

```
# В этом примере разрешено создание секрета по пути "secret/foo"
# с любыми параметрами. При этом параметр "bar" может принимать
# значения только "zip" или "zap"
path "secret/foo" {
    capabilities = ["create"]
    allowed_parameters = {
        "bar" = ["zip", "zap"]
        "*" = []
    }
}
```

- denied\_parameters** - Список ключей и значений, которые не разрешены на данном пути. Любые значения, указанные здесь, имеют приоритет над `allowed_parameters`. Возможны различные способы использования данного поля:

- Указание пустого списка ( [] ) запрещает любые изменения указанного параметра:

```
# В примере разрешено обновление всех пользователей в userpass.
# При этом запрещено изменять значения параметров "token_policies" и
# "policies"
path "auth/userpass/users/*" {
    capabilities = ["update"]
    denied_parameters = {
        "token_policies" = []
        "policies" = []
    }
}
```

- Указание списка значений запрещает установку этих значений для указанного параметра:

```
# В примере разрешено создание и обновление любых ролей во
# встроенном методе аутентификации.
# При этом параметр "allowed_policies" не может принимать значение
# "admin"
path "auth/token/roles/*" {
    capabilities = ["create", "update"]
    denied_parameters = {
        "allowed_policies" = ["admin"]
    }
}
```

- Указание "\*" с пустым списком запрещает любые параметры:

```
# В примере разрешено создание и изменение транзитных ключей
# При этом нельзя задавать никакие параметры.
path "transit/keys/*" {
    capabilities = ["create", "update"]
    denied_parameters = {
        "*" = []
    }
}
```

- Если указаны какие-либо параметры, то все неуказанные параметры разрешены, если только не задано значение `allowed_parameters`, в этом случае применяются обычные правила.

Значения параметров также поддерживают глоббинг префиксов/суффиксов.

Оглоблирование включается путем добавления к значению префикса (\*):

```
# Для параметра "bar" разрешено устанавливать только значения, начинающиеся с
# "foo-".
path "secret/foo" {
    capabilities = ["create"]
    allowed_parameters = {
        "bar" = ["foo-*"]
    }
}
```

## 6. Встроенные политики

В StarVault есть две встроенные политики: **default** и **root**.

### 6.1. Политика default

Политика **default** - это встроенная политика StarVault, которую нельзя удалить. По умолчанию она прикрепляется ко всем токенам, но может быть явно исключена во время создания токена с помощью поддерживаемых методов аутентификации.

Политика содержит базовую функциональность, такую как возможность для токена просматривать данные о себе и использовать данные своего вложенного хранилища (cubby-hole). Однако StarVault не фиксирует ее содержание. Ее можно изменять в соответствии с вашими потребностями. StarVault никогда не перезаписывает внесенные изменения.

Чтобы просмотреть все разрешения, предоставленные политикой по умолчанию для вашей установки StarVault, выполните команду:

```
starvault policy read default
```

Чтобы отключить прикрепление политики **default** при создании токена используйте опцию `--no-default-policy` или `token_no_default_policy` (в зависимости от метода аутентификации). Например:

```
starvault token create --no-default-policy
```

## 6.2. Политика root

Политика **root** - это встроенная политика StarVault, которую нельзя изменить или удалить. Любой пользователь, связанный с этой политикой, становится пользователем **root**. Пользователь **root** может делать все, что угодно в StarVault. Поэтому настоятельно рекомендуется отзывать все root-токены перед запуском StarVault в продуктив.

При первой инициализации сервера StarVault всегда существует один пользователь **root**. Этот пользователь используется для начальной настройки и установки StarVault. После настройки первоначальный корневой токен должен быть отозван, и следует использовать более строго контролируемых пользователей и аутентификацию.

## 7. Управление политиками

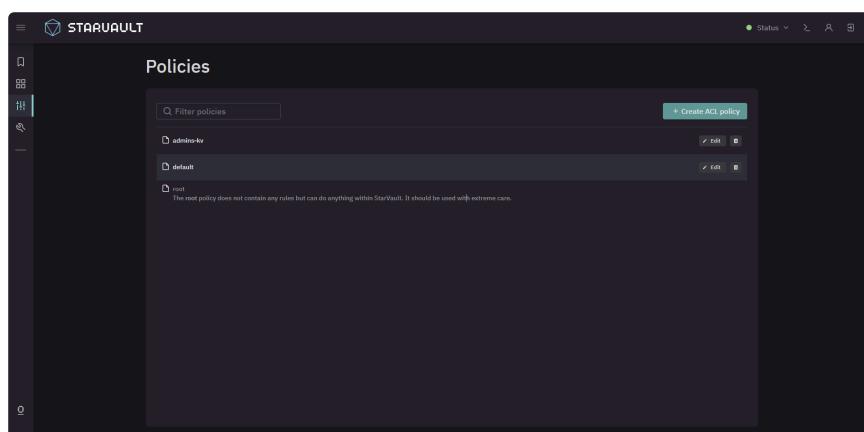
Политики создаются в выбранном вами редакторе. Они могут быть созданы в формате HCL или JSON, а синтаксис подробно описан выше. После сохранения политики должны быть загружены в StarVault, прежде чем их можно будет использовать.

### 7.1. Просмотр списка политик

Для просмотра списка зарегистрированных политик можно использовать UI и CLI.

#### UI

1. Аутентифицируйтесь в пользовательском интерфейсе с достаточными правами.
2. Перейдите на вкладку **Policies**.



## CLI

1. Аутентифицируйтесь с достаточными правами с помощью команды `starvault login`.
2. Введите любую из следующих команд:

```
starvault read sys/policy

Key          Value
---          -----
keys         [admins-kv default root]
policies     [admins-kv default root]

starvault policy list

admins-kv
default
root
```



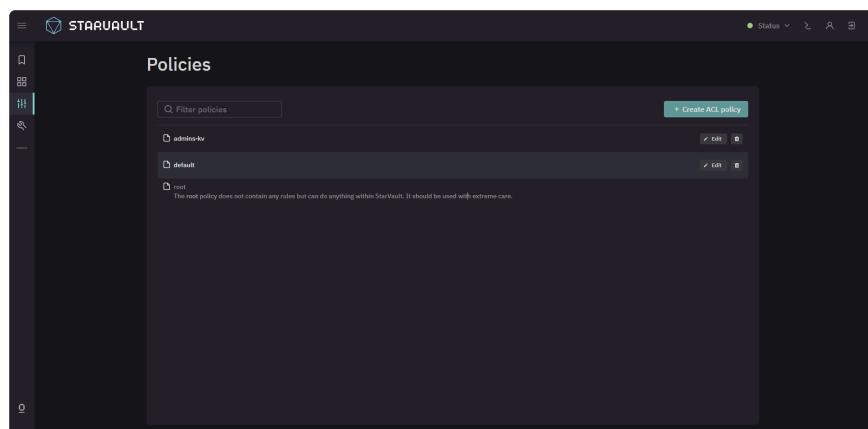
Обратите внимание на различия в выводе.

## 7.2. Создание политик

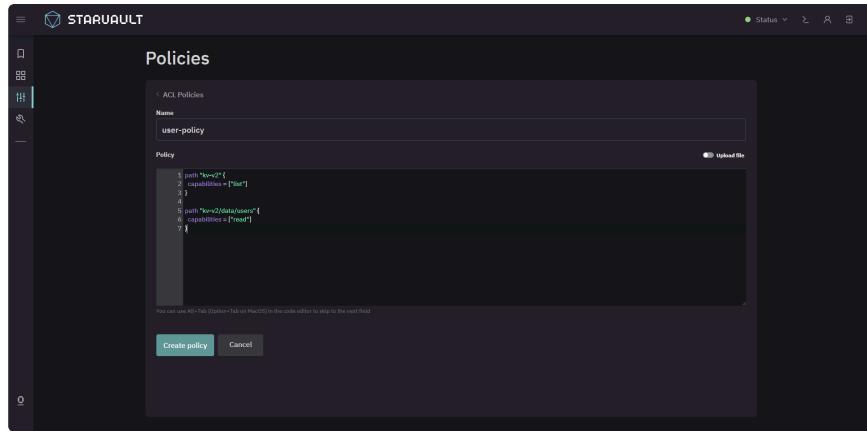
Для создания политик можно использовать UI и CLI.

### UI

1. Аутентифицируйтесь в пользовательском интерфейсе с достаточными правами.
2. Перейдите на вкладку **Policies**.



3. Нажмите [**Create ACL policy**].
4. Введите имя и конфигурацию политики.



5. Нажмите [ **Create policy** ].

## CLI

1. Аутентифицируйтесь с достаточными правами с помощью команды `starvault login`.
2. С помощью любого доступного редактора создайте файл описания политики.

Например:

```
vim admin-policy.hcl
```

Содержимое:

```
path "kv-v2/*" {
    capabilities = ["create", "update", "read", "patch", "list"]
}
```

3. Создайте политику из файла описания с помощью команды:

```
starvault policy write <policy-name> <policy-file.hcl> ① ②
```

- ① <policy-name> - имя создаваемой политики  
② <policy-file.hcl> - путь к файлу описания

Например:

```
starvault policy write admin-policy admin-policy.hcl
```

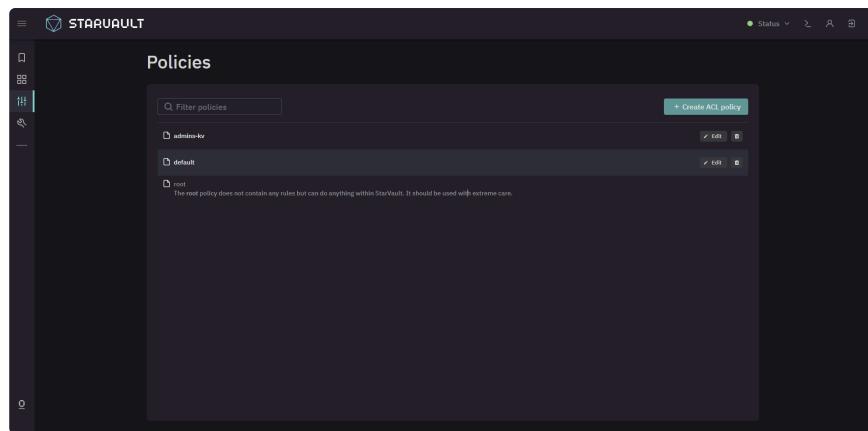
## 7.3. Обновление политик

Существующие политики могут быть обновлены для изменения разрешений. Для обновления политик можно использовать UI и CLI.

## UI

1. Аутентифицируйтесь в пользовательском интерфейсе с достаточными правами.

2. Перейдите на вкладку **Policies**.



3. В строке нужной политики нажмите [ **Edit** ].

4. Внесите необходимые изменения в конфигурацию

5. Нажмите [ **Save** ].

### **CLI**

1. Аутентифицируйтесь с достаточными правами с помощью команды `starvault login`.

2. С помощью любого доступного редактора создайте или отредактируйте файл описания политики. Например:

```
vim admin-policy.hcl
```

Содержимое:

```
path "kv-v2/*" {
    capabilities = ["create", "update", "read", "patch", "list", "delete"]
}
```

3. Выполните команду для обновления политики из файла описания (команда аналогична созданию):

```
starvault policy write <policy-name> <policy-file.hcl> ① ②
```

① <policy-name> - имя политики, которую необходимо обновить

② <policy-file.hcl> - путь к файлу описания

Например:

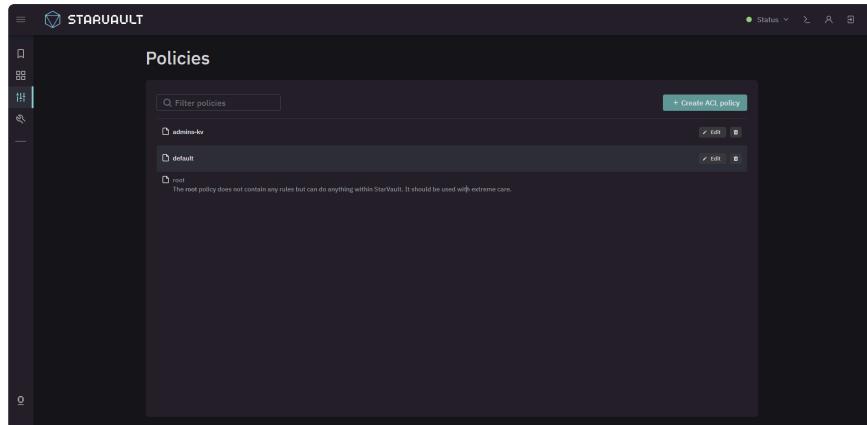
```
starvault policy write admin-policy admin-policy.hcl
```

## **7.4. Удаление политик**

Существующие политики могут быть удалены (за исключением **default** и **root**). Для удаления политик можно использовать UI и CLI.

## UI

1. Аутентифицируйтесь в пользовательском интерфейсе с достаточными правами.
2. Перейдите на вкладку **Policies**.



3. В строке нужной политики нажмите или нажмите на нужную политику для перехода в подробное представление и нажмите [ **Delete** ].
4. Подтвердите удаление, нажав [ **Delete** ]

## CLI

1. Аутентифицируйтесь с достаточными правами с помощью команды `starvault login`.
2. Выполните любую из следующих команд для удаления политики:

```
starvault policy delete <policy-name> ①
```

① <policy-name> - имя политики, которую необходимо удалить

```
starvault delete sys/policy/<policy-name> ①
```

① <policy-name> - имя политики, которую необходимо удалить

Например:

```
starvault policy delete admin-policy
```

## 8. Ассоциация политик

StarVault может автоматически связать набор политик с токеном. Эта конфигурация значительно отличается в разных методах аутентификации. Для простоты в этом примере будет использован встроенный в StarVault метод аутентификации **userpass**:

```
starvault write auth/userpass/users/new-user \ ①  
    password=s3cr3t! \  
    token_policies=user-policy ②
```

① Создается пользователь с именем **new-user**

② Токен пользователя связывается с политикой **user-policy**

## 9. Токены

У токенов есть два набора политик:

- политики идентификации, которые вычисляются на основе сущности и ее групп;
- политики токена, которые либо определяются на основе метода входа в систему, либо, в случае явного создания токена через API, являются входными данными для создания токена.

Все нижеследующее касается исключительно политик токенов: политиками идентификации токена нельзя управлять, кроме как изменяя базовые сущности, группы и членство в группах.

Токены ассоциируются со своими политиками во время создания. Например:

```
starvault token create -policy=dev-readonly -policy=logs
```

Обычно можно указывать только те политики, которые присутствуют в политиках текущего токена (т.е. родителя нового токена). Однако пользователи **root** могут назначать любые политики.

Не существует способа изменить политики, связанные с токеном, после того как токен был выпущен. Для получения нового набора политик токен должен быть отозван и получен новый.

Однако содержимое политик анализируется в режиме реального времени при каждом использовании токена. В результате, если политика была изменена, измененные правила будут действовать в следующий раз, когда токен с этой политикой будет использован для вызова StarVault.

# Аутентификация на основе токенов

Метод аутентификации с помощью токена является встроенным и является основным для клиента. Для аутентификации могут использоваться и другие методы аутентификации, но в конечном итоге они приводят к созданию клиентского токена, управляемого серверной частью токена.

Каждый токен обладает рядом свойств:

- ID - основной идентификатор токена - это случайно сгенерированное значение
- Display Name - необязательно, отображаемое имя;
- Metadata - метаданные, используемые для ведения журнала аудита;
- Number of Uses - необязательно, ограничение на количество использования;
- Parent ID - необязательно, родительский токен, который создал текущий дочерний токен;
- Policies - связанный список политик ACL;
- Source Path - путь, по которому был сгенерирован токен (например, auth/ldap/login ).

Свойства токена остаются неизменными после его создания. Исключением из этого правила является количество использований, которое уменьшается при каждом запросе. Каждое из этих свойств позволяет StarVault выполнять ряд полезных функций.

Каждый токен содержит исходный путь или путь входа в систему, который использовался для создания токена. Это используется для разрешения отзыва на основе исходного кода. Например, если мы считаем, что наша организация на LDAP была скомпрометирована, мы можем отозвать все токены, сгенерированные с помощью auth/ldap/login . Это можно сделать, используя sys/revoke-prefix/ API с префиксом auth/ldap/ . Отмена префикса приведет к аннулированию всех клиентских токенов, сгенерированных по этому пути, а также всех динамических секретов, сгенерированных этими токенами. Это обеспечивает эффективную процедуру быстрого реагирования во время потенциальной компрометации.

Если токен создан другим методом аутентификации, у него нет родительского токена. Однако у любых токенов, созданных с помощью API auth/token/create , есть родительский токен, а именно токен, используемый для выполнения этого запроса. Поддерживая эту связь между родителями и дочерними элементами, StarVault моделирует деревья токенов. Дочерние токены могут быть созданы с использованием подмножества родительских политик, что позволяет отказаться от привилегий. Когда токен отзывается, вместе с ним отзывается и все поддерево токенов. Это позволяет клиентам безопасно генерировать дочерние токены, а затем отзывать их все вместе с корневым.

Дочерние токены очень полезны, особенно в сочетании с токенами ограниченного использования. При создании токена можно дополнительно указать количество его использований. Если количество использований равно единице, токен будет использоваться один раз. Это означает, что токен может быть использован для одного запроса, прежде чем будет автоматически отозван. Это может быть распространено на любое количество применений. Токены ограниченного использования нельзя использовать для создания дополнительных токенов, но они могут стать эффективным способом обеспечения крайне ограниченного доступа к StarVault.

---

# Блокировка пользователей

Если пользователь несколько раз подряд вводит неверные учетные данные, StarVault на некоторое время прекращает попытки проверки их учетных данных и вместо этого немедленно возвращается с ошибкой "отказано в разрешении". Это "блокировка пользователя". Время, на которое пользователь будет заблокирован, называется "продолжительностью блокировки". Пользователь сможет войти в систему по истечении этого срока. Количество неудачных попыток входа, после которых пользователь будет заблокирован, называется "порогом блокировки". Счетчик порога блокировки обнуляется через несколько минут без попыток входа в систему или после успешной попытки входа в систему. Время, по истечении которого счетчик будет обнулен после отсутствия попыток входа в систему, называется "сброс счетчика блокировки". Это может предотвратить как автоматические, так и целевые запросы, т.е. атаки на основе подбора пароля пользователем, а также автоматические атаки.

Функция блокировки пользователя включена по умолчанию. Значения по умолчанию для "порога блокировки" - 5 попыток, "продолжительности блокировки" - 15 минут, "сброса счетчика блокировки" - 15 минут.

Функцию блокировки пользователя можно отключить следующим образом:

- Это можно отключить глобально, используя переменную окружения `VAULT_DISABLE_USER_LOCKOUT`.
- Его можно отключить для всех поддерживающих методов аутентификации (`ldap`, `user pass` и `approve`) или для конкретного поддерживающего метода аутентификации, используя параметр `disable_lockout` в разделе `user_lockout` в файле конфигурации. Для получения более подробной информациисмотрите конфигурацию блокировки пользователя.
- Его можно отключить для определенного режима авторизации, используя `auth tune`. Пожалуйста, смотрите команду `auth tune` или `auth tune api` для получения более подробной информации.



Эта функция поддерживается только методами аутентификации `user pass`, `ldap` и `approve`.

## 1. Приоритет

Приоритет настройки блокировки пользователя следующий:

1. Настройка для автоматического подключения с помощью `tune`

2. Настройка метода аутентификации в файле конфигурации
3. Настройка для "всех" методов аутентификации в файле конфигурации
4. Значения по умолчанию

Приоритет отключения блокировки пользователя следующий:

1. Отключить с помощью переменной среды VAULT\_DISABLE\_USER\_LOCKOUT
2. Настройка для автоматического подключения с помощью tune
3. Настройка для метода аутентификации в конфигурационном файле
4. Настройка для "всех" методов аутентификации в конфигурационном файле
5. Значения по умолчанию

## 2. Конфигурация

---

Параметры блокировки пользователя можно настроить с помощью конфигурационного файла для всех методов авторизации или для конкретного метода авторизации (user pass, ldap или approle). Пожалуйста, посмотрите конфигурацию блокировки пользователя для получения более подробной информации.

Конфигурацию блокировки пользователя для метода auth по заданному пути можно настроить с помощью auth tune . Пожалуйста, посмотрите команду auth tune или auth tune api для получения более подробной информации.

## 3. API

---

Пожалуйста, смотрите sys/locked-users API для получения более подробной информации.