Инструкция по работе с движком KV

ЦУП2.0 модуль управления секретами

SecretManagement

Table of Contents

Механизм KV секретов используется для хранения произвольных секретов в настроенном физическом хранилище Vault.

Запись ключа в бэкэнд заменит старое значение.

Имена ключей всегда должны быть строками. Если вы записываете нестроковые значения напрямую через CLI, они будут преобразованы в строки. Однако вы можете сохранить нестроковые значения, записав пары ключ/значение в Vault из файла JSON или используя HTTP API.

Этот механизм секретов учитывает различие между возможностями *create* и *update* внутри политик ACL.

1 KV Secrets Engine - Версия 1

1.1 Настройка

Чтобы включить хранилище kv версии 1:

```
vault secrets enable -version=1 kv
```

1.2 Применение

После того, как механизм секретов настроен и пользователь/компьютер имеет токен Vault с соответствующими разрешениями, он может генерировать учетные данные. Механизм kv секретов позволяет записывать ключи с произвольными значениями.

1.Записать произвольные данные:

```
$ vault kv put kv/my-secret my-value=s3cr3t Success! Data written to:
kv/my-secret
```

2.Чтение произвольных данных:

```
$ vault kv get kv/my-secret

Key          Value

---           -----

my-value     s3cr3t
```

3.Перечисление ключей

```
 vault kv list kv/ Keys----

my-secret
```

4.Удаление ключа

```
$ vault kv delete kv/my-secret Success! Data deleted (if it existed) at:
kv/my-secret
```

1.3 TTL

В отличие от других механизмов секретов, механизм секретов KV не применяет TTL для истечения срока действия. Вместо этого *lease_duration* это подсказка о том, как часто потребители должны проверять новое значение.

Если предоставлен ключ *ttl*, механизм секретов KV будет использовать это значение в качестве продолжительности времени аренды:

```
 $ vault kv put kv/my-secret ttl=30m my-value=s3cr3t Success! Data written
to: kv/my-secret
```

Даже с установленным и настроенным *ttl,* механизмом секретов *никогда не* удаляет данные самостоятельно. *ttl* ключ просто консультативный .

При чтении значения с ttl и ttl ключом, и интервалом обновления, будет отражать значение:

```
vault kv get kv/my-secret

Key             Value

---             -----

my-value        s3cr3t

ttl              30m
```

1.4 API

1.4.1 Чтение секретов

Endpoint получает секрет в указанном месте.

Method	Path
GET	/secret/:path

Параметры

path (string: <required>) – Указывает путь секрета для чтения. Это указывается как часть URL-адреса.

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    https://127.0.0.1:8200/v1/secret/my-secret
```

Пример ответа

```
{
  "auth": null,
  "data": {
    "foo": "bar",
    "ttl": "1h"
  },
  "lease_duration": 3600,
  "lease_id": "",
  "renewable": false
}
```

1.4.2 Список секретов

Endpoint возвращает список имен ключей в указанном месте.

Method	Path
`LIST`	`/secret/:path`

Параметры

path (string: <required>) - Указывает путь секретов для списка. Это указывается как часть URL-адреса.

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    --request LIST \
    https://127.0.0.1:8200/v1/secret/my-secret
```

Пример ответа

```
{
  "auth": null,
  "data": {
    "keys": ["foo", "foo/"]
  },
  "lease_duration": 2764800,
  "lease_id": "",
  "renewable": false
}
```

1.4.3 Создание/обновление секрета

Endpoint хранит секрет в указанном месте. Если значение еще не существует, вызывающий маркер должен иметь политику ACL, предоставляющую возможность создания. Если значение уже существует, вызывающий маркер должен иметь политику ACL, предоставляющую возможность обновления.

Method	Path
`POST`	`/secret/:path`

Параметры

- path (string: <required>) – Указывает путь к секретам для создания/обновления. Это указывается как часть URL-адреса.
- :key (string: "") – Указывает ключ в паре со связанным значением, которое должно храниться в данном месте. Можно указать несколько пар ключ/значение, и все они будут возвращены при операции чтения. Ключ с именем ttl будет вызывать особое поведение.

Пример

```
{
  "foo": "bar",
  "zip": "zap"
}
```

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    https://127.0.0.1:8200/v1/secret/my-secret
```

1.4.4 Удаление секрета

Endpoint удаляет секрет в указанном месте.

Method	Path
`DELETE`	`/secret/:path`

Параметры

path (string: <required>) – Указывает путь к удаляемому секрету. Это указывается как часть URL-адреса.

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    --request DELETE \
    https://127.0.0.1:8200/v1/secret/my-secret
```

2 KV Secrets Engine - Версия 2

2.1 Настройка

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

Движок секретов KV-v2 можно включить следующим образом:

```
$ vault secrets enable -version=2 kv
```

Или вы можете KV-v2 указать тип механизма секретов:

```
vault secrets
 enable kv-v2
```

Кроме того, при запуске сервера в режиме разработки механизм секретов KVv2 включен по умолчанию в пути `secret/` (для серверов без разработки это в настоящее время v1). Его можно отключать, перемещать или включать несколько раз по разным путям. Каждый экземпляр движка KV secrets изолирован и уникален.

2.2 Обновление с версии 1

Существующее хранилище kv версии 1 можно обновить до хранилища kv версии 2 через интерфейс командной строки или API, как показано ниже. Это запустит процесс обновления для обновления существующих данных ключа/значения до версионного формата. Во время этого процесса mount будет недоступен. Этот процесс может занять много времени, поэтому планируйте его соответствующим образом.

После обновления до версии 2 прежних путей, по которым были доступны данные, больше не будет достаточно. Вам потребуется настроить пользовательские политики, чтобы добавить доступ к путям версии 2, как подробно описано в разделе « Правила ACL» ниже . Точно так же пользователям/приложениям потребуется обновить пути, по которым они взаимодействуют с данными kv после их обновления до версии 2.

Существующую версию 1 kv можно обновить до KV версии 2 с помощью команды CLI:

```
$ vault kv enable-versioning secret/

Success! Tuned the secrets engine at: secret/
```

или через API:

```
$ cat payload.json
{
  "options": {
      "version": "2"
  }
}

 curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/sys/mounts/secret/tune
```

2.3 Правила ACL

Хранилище kv версии 2 использует API с префиксом, который отличается от API версии 1. Перед обновлением с версии 1 необходимо изменить правила ACL. Также разные пути в API версии 2 могут быть по-разному составлены ACL.

Версии для записи и чтения имеют префикс `data/`пути. Эта политика, которая работала для версии 1:

```
path "secret/dev/team-1/*" {
  capabilities = ["create", "update", "read"]
}
```

Следует изменить на:

```
path "secret/data/dev/team-1/*" {
  capabilities = ["create", "update", "read"]
}
```

Для этого бэкенда существуют разные уровни удаления данных. Чтобы предоставить политике права на удаление последней версии ключа:

```
path "secret/data/dev/team-1/*" {
  capabilities = ["delete"]
}
```

Чтобы разрешить политике удалять любую версию ключа:

```
path "secret/delete/dev/team-1/*" { capabilities = ["update"] }
```

Чтобы разрешить политике восстановить данные:

```
path "secret/undelete/dev/team-1/*" { capabilities = ["update"] }
```

Чтобы разрешить политике уничтожать версии:

```
path "secret/destroy/dev/team-1/*" { capabilities = ["update"] }
```

Чтобы разрешить политике отображать ключи:

```
path "secret/metadata/dev/team-1/*" { capabilities = ["list"] }
```

Чтобы разрешить политике просматривать метаданные для каждой версии:

```
path "secret/metadata/dev/team-1/*" { capabilities = ["read"] }
```

Чтобы разрешить политике безвозвратное удаление всех версий и метаданных ключа:

```
path "secret/metadata/dev/team-1/*" { capabilities = ["delete"] }
```

Поля `allowed_parameters`, `denied_parameters`и `required_parameters` не поддерживаются для политик, используемых с хранилищем kv версии 2. Описание этих параметров см. в разделе « Концепции политик».

Дополнительную информацию см. в Спецификации API .

2.4 Применение

После того, как механизм секретов настроен и пользователь/компьютер имеет токен Vault с соответствующими разрешениями, он может генерировать учетные данные. Механизм `KV` секретов позволяет записывать ключи с произвольными значениями.

Синтаксис пути KV-v1 для ссылки на секрет (`secret/foo`) по-прежнему может использоваться в KV-v2, но рекомендуется использовать `-mount=secret` синтаксис флага, чтобы избежать путаницы с фактическим путем к секрету (`secret/data/foo`является реальным путем).

Запись/чтение произвольных данных

1.Запись произвольных данных:

```
$ vault kv put -mount=secret my-secret foo=a bar=b
Key              Value
---              -----
created_time     2019-06-19T17:20:22.985303Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          1
```

2. Чтение произвольных данных:

```
$ vault kv get -mount=secret my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2019-06-19T17:20:22.985303Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          1

====== Data ======
Key         Value
---         -----
foo         a
bar         b
```

3.При записи другой версии, предыдущая версия все равно будет доступна. Флаг `-cas` можно дополнительно передать для выполнения операции проверки и установки. Если не установлено, запись будет разрешена. Для успешной записи необходимо установить текущую версию секрета. Если установлено значение 0, запись будет разрешена только в том случае, если ключ не существует, поскольку неустановленные ключи не имеют

никакой информации о версии. Также помните, что обратимое удаление не удаляет какие-либо базовые данные версии из хранилища. Для записи в обратимо удаленный ключ параметр cas должен соответствовать текущей версии ключа.

```
$ vault kv put -mount=secret -cas=1 my-secret foo=aa bar=bb
Key              Value
---              -----
created_time     2019-06-19T17:22:23.369372Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          2
```

4.При чтении сейчас будет возвращена самая новая версия данных:

```
$ vault kv get -mount=secret my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2019-06-19T17:22:23.369372Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          2

====== Data ======
Key         Value
---         -----
foo         aa
bar         bb
```

5.Частичные обновления можно выполнить с помощью команды `vault kv patch`. Сначала команда попытается выполнить HTTP `PATCH`-запрос, для которого требуется `patch` возможность ACL. Запрос `PATCH` завершится ошибкой, если используемый токен связан с политикой, которая не содержит `patch` возможности. В этом случае команда выполнит чтение, локальное обновление и последующую запись, для чего требуются как возможности ACL, так `read` и `update` ACL.

Флаг `-cas` можно дополнительно передать для выполнения операции проверки и установки. Он будет использоваться только в случае первоначального `PATCH` запроса. Поток чтения и записи будет использовать `version` значение из секрета, возвращенного чтением, для выполнения операции проверки и установки при последующей записи.

```
$ vault kv patch -mount=secret -cas=2 my-secret bar=bbb
Key              Value
---              -----
created_time     2019-06-19T17:23:49.199802Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          3
```

6.Команда `vault kv patch` также поддерживает `-method` флаг, который можно использовать для указания HTTP `PATCH` или чтения и записи. Поддерживаемые значения: `patch`и `rw` для HTTP `PATCH` и чтение-затем-запись соответственно.

Выполните патч, используя `patch` метод:

```
$ vault kv patch -mount=secret -method=patch -cas=2 my-secret bar=bbb
Key              Value
---              -----
created_time     2019-06-19T17:23:49.199802Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          3
```

Выполните патч, используя метод чтения и записи:

```
$ vault kv patch -mount=secret -method=rw my-secret bar=bbb
Key              Value
---              -----
created_time     2019-06-19T17:23:49.199802Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          3
```

7.Чтение после исправления вернет самую новую версию данных, в которой были обновлены только указанные поля:

```
$ vault kv get -mount=secret my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2019-06-19T17:23:49.199802Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          3

====== Data ======
Key         Value
---         -----
foo         aa
bar         bbb
```

8.Доступ к предыдущим версиям можно получить с помощью `-version` флага:

```
$ vault kv get -mount=secret -version=1 my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2019-06-19T17:20:22.985303Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          1

====== Data ======
Key         Value
---         -----
foo         a
bar         b
```

Удаление и уничтожение данных

При удалении данных стандартная команда `vault kv delete` выполнит обратимое удаление. Он пометит версию как удаленную и заполнит

отметку `deletion_time` времени. Обратимое удаление не удаляет базовые данные версии из хранилища, что позволяет восстановить версию. Команда `vault kv undelete` обрабатывает восстановление версий.

Данные версии удаляются навсегда только в том случае, если у ключа больше версий, чем разрешено настройкой max-versions, или при использовании `vault kv destroy`. При использовании команды уничтожения данные базовой версии будут удалены, а метаданные ключа будут помечены как уничтоженные. Если версия очищается путем перехода к max-versions, метаданные версии также будут удалены из ключа.

См. приведенные ниже команды для получения дополнительной информации:

1.Последнюю версию ключа можно удалить с помощью команды удаления, для этого также требуется `-versions` флаг для удаления предыдущих версий:

```
$ vault kv delete -mount=secret my-secret
Success! Data deleted (if it existed) at: secret/data/my-secret
```

2.Версии можно восстановить:

```
$ vault kv undelete -mount=secret -versions=2 my-secret
Success! Data written to: secret/undelete/my-secret

 vault kv get -mount=secret my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2019-06-19T17:23:21.834403Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          2

====== Data ======
Key         Value
---         -----
my-value    short-lived-s3cr3t
```

3.Уничтожение версии безвозвратно удаляет базовые данные:

```
$ vault kv destroy -mount=secret -versions=2 my-secret
Success! Data written to: secret/destroy/my-secret
```

Ключевые метаданные

Все версии и ключевые метаданные можно отслеживать с помощью команды метаданных и API. Удаление ключа метаданных приведет к безвозвратному удалению всех метаданных и версий этого ключа.

См. приведенные ниже команды для получения дополнительной информации:

1.Все метаданные и версии ключа можно просмотреть:

```
$ vault kv metadata get -mount=secret my-secret
========== Metadata ==========
Key                     Value
---                     -----
cas_required            false
created_time            2019-06-19T17:20:22.985303Z
current_version         2
custom_metadata         <nil>
delete_version_after    0s
max_versions            0
oldest_version          0
updated_time            2019-06-19T17:22:23.369372Z

====== Version 1 ======
Key              Value
---              -----
created_time     2019-06-19T17:20:22.985303Z
deletion_time    n/a
destroyed        false

====== Version 2 ======
Key              Value
---              -----
created_time     2019-06-19T17:22:23.369372Z
deletion_time    n/a
destroyed        true
```

2.Параметры метаданных для ключа можно настроить:

```
$ vault kv metadata put -mount=secret -max-versions 2 -delete-version-
after="3h25m19s" my-secret
Success! Data written to: secret/metadata/my-secret
```

Настройки удаления версии после будут применяться только к новым версиям. Изменения максимальных версий будут применены при следующей записи:

```
$ vault kv put -mount=secret my-secret my-value=newer-s3cr3t
Key              Value
---              -----
created_time     2019-06-19T17:31:16.662563Z
custom_metadata  <nil>
deletion_time    2019-06-19T20:56:35.662563Z
destroyed        false
version          4
```

Когда у ключа больше версий, чем max, самые старые версии очищаются:

```
$ vault kv metadata get -mount=secret my-secret
========== Metadata ==========
Key                     Value
---                     -----
cas_required            false
created_time            2019-06-19T17:20:22.985303Z
current_version         4
custom_metadata         <nil>
delete_version_after    3h25m19s
max_versions            2
oldest_version          3
updated_time            2019-06-19T17:31:16.662563Z

====== Version 3 ======
Key              Value
---              -----
created_time     2019-06-19T17:23:21.834403Z
deletion_time    n/a
destroyed        true

====== Version 4 ======
Key              Value
---              -----
created_time     2019-06-19T17:31:16.662563Z
deletion_time    2019-06-19T20:56:35.662563Z
destroyed        false
```

Метаданные ключа секрета могут содержать пользовательские метаданные, используемые для описания секрета. Данные будут храниться в виде пар ключ-значение "строка-строка". Флаг `-custom-metadata` можно повторить, чтобы добавить несколько пар ключ-значение.

Команду `vault kv metadata put` можно использовать для полной перезаписи значения `custom_metadata`:

```
$ vault kv metadata put -mount=secret -custom-metadata=foo=abc -custom-
metadata=bar=123 my-secret
Success! Data written to: secret/metadata/my-secret

 vault kv get -mount=secret my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2019-06-19T17:22:23.369372Z
custom_metadata  map[bar:123 foo:abc]
deletion_time    n/a
destroyed        false
version          2

====== Data ======
Key         Value
---         -----
foo         aa
bar         bb
```

Команду `vault kv metadata patch` можно использовать для частичной перезаписи значения `custom_metadata`. Следующий вызов `custom_metadata` обновит подполе , `foo` но оставит его `bar` нетронутым:

```
$ vault kv metadata patch -mount=secret -custom-metadata=foo=def my-secret
Success! Data written to: secret/metadata/my-secret

 vault kv get -mount=secret my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2019-06-19T17:22:23.369372Z
custom_metadata  map[bar:123 foo:def]
deletion_time    n/a
destroyed        false
version          2

====== Data ======
Key         Value
---         -----
foo         aa
bar         bb
```

3.Навсегда удалить все метаданные и версии для ключа:

```
$ vault kv metadata delete -mount=secret my-secret
Success! Data deleted (if it existed) at: secret/metadata/my-secret
```

2.5 API

2.5.1 Конфигурация KV движка

Этот путь настраивает параметры серверного уровня, которые применяются к каждому ключу в хранилище 'ключ-значение'.

Method	Path
`POST`	`/secret/config`

Параметры

- max_versions (int: 0) – Количество версий для каждого ключа. Это значение применяется ко всем ключам, но параметр метаданных ключа может перезаписать это значение. Как только у ключа будет больше настроенных разрешенных версий, самая старая версия будет безвозвратно удалена. Если используется 0 или значение не задано, Vault сохранит 10 версий.
- cas_required (bool: false) – Если true, все ключи потребуют установки параметра cas во всех запросах на запись.
- delete_version_after (string:"0s") – Если установлено, указывает продолжительность времени до удаления версии.

Пример

```
{
  "max_versions": 5,
  "cas_required": false,
  "delete_version_after": "3h25m19s"
}
```

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    https://127.0.0.1:8200/v1/secret/config
```

2.5.2 Чтение конфигурации KV движка

Этот путь извлекает текущую конфигурацию для серверной части секретов по заданному пути.

Method	Path
GET	/secret/config

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    https://127.0.0.1:8200/v1/secret/config
```

Пример ответа

```
{
  "data": {
    "cas_required": false,
    "delete_version_after": "3h25m19s",
    "max_versions": 0
  }
}
```

2.5.3 Чтение версии секрета

Endpoint получает секрет в указанном месте. Поля метаданных created_time, deletion_time, destroy и version зависят от версии. Поле custom_metadata является частью метаданных ключа секрета и включается в ответ независимо от того, имеет ли вызывающий токен доступ для чтения к связанной конечной точке метаданных.

Method	Path
GET	/secret/data/:path?version=:version-number

Параметры

- path (string: <required>) – Указывает путь секрета для чтения. Это указывается как часть URL-адреса.
- version (int: 0) - Указывает возвращаемую версию. Если не установлено, возвращается последняя версия.

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    https://127.0.0.1:8200/v1/secret/data/my-secret?version=2
```

Пример ответа

```
{
  "data": {
    "data": {
      "foo": "bar"
    },
    "metadata": {
      "created_time": "2018-03-22T02:24:06.945319214Z",
      "custom_metadata": {
        "owner": "jdoe",
        "mission_critical": "false"
      },
      "deletion_time": "",
      "destroyed": false,
      "version": 2
    }
  }
}
```

2.5.4 Создание/обновление секретов

Endpoint создает новую версию секрета в указанном месте. Если значение еще не существует, вызывающий маркер должен иметь политику ACL, предоставляющую возможность создания. Если значение уже существует, вызывающий маркер должен иметь политику ACL, предоставляющую возможность обновления.

Method	Path
`POST`	`/secret/data/:path`

Параметры

- options (Map: <optional>) – Объект, содержащий настройки параметров.
- cas (int: <optional>) - Этот флаг обязателен, если для cas_required установлено значение true либо в секрете, либо в конфигурации движка. Если не установлено, запись будет разрешена. Чтобы запись была успешной, в cas должна быть установлена текущая версия секрета. Если установлено значение 0, запись будет разрешена только в том случае, если ключ не существует, поскольку неустановленные ключи не имеют никакой информации о версии. Также помните, что обратимое удаление не удаляет какие-либо базовые данные версии из хранилища. Для записи в обратимо удаленный ключ параметр cas должен соответствовать текущей версии ключа.
- data (Map: <required>) – Содержимое карты данных будет сохранено и возвращено при чтении.

Пример

```
{
  "options": {
    "cas": 0
  },
  "data": {
    "foo": "bar",
    "zip": "zap"
  }
}
```

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    https://127.0.0.1:8200/v1/secret/data/my-secret
```

Пример ответа

```
{
  "data": {
    "created_time": "2018-03-22T02:36:43.986212308Z",
    "custom_metadata": {
      "owner": "jdoe",
      "mission_critical": "false"
    },
    "deletion_time": "",
    "destroyed": false,
    "version": 1
  }
}
```

2.5.5 Патч секрета

Endpoint предоставляет возможность исправить существующий секрет в указанном месте. Секрет нельзя ни удалять, ни уничтожать. Вызывающий токен должен иметь политику ACL, предоставляющую возможность исправления. В настоящее время поддерживается только исправление слияния JSON, которое должно быть указано с использованием значения заголовка Content-Type application/merge-patch+json. Новая версия будет создана после успешного применения исправления с предоставленными данными.

Method	Path
`PATCH`	`/secret/data/:path`

Параметры

- options (Map: <optional>) – Объект, содержащий настройки параметров.
- cas (int: <optional>) - Этот флаг обязателен, если для cas_required установлено значение true либо в секрете, либо в конфигурации движка. Чтобы запись была успешной, в cas должна быть установлена текущая версия секрета. Необходимо выполнить операцию исправления для существующего ключа, поэтому указанное значение cas должно быть больше 0.
- data (Map: <required>) – Содержимое карты данных будет применено как частичное обновление к существующей записи через патч слияния JSON для существующей записи.

Пример

```
{
  "options": {
    "cas": 1
  },
  "data": {
    "foo": "a",
    "bar": {
      "baz": "b"
    }
  }
}
```

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    --header "Content-Type: application/merge-patch+json"
    --request PATCH \
    --data @payload.json \
    https://127.0.0.1:8200/v1/secret/data/my-secret
```

Пример ответа

```
{
  "data": {
    "created_time": "2021-09-10T15:26:08.684999Z",
    "custom_metadata": {
      "owner": "jdoe",
      "mission_critical": "false"
    },
    "deletion_time": "",
    "destroyed": false,
    "version": 2
  }
}
```

2.5.6 Чтение подраздела секретов

Endpoint предоставляет подразделы в секретной записи, которая существует по запрошенному пути. Секретная запись по этому пути будет получена и лишена всех данных путем замены базовых значений конечных ключей (т. е. ключей без карты или ключей карты без базовых подключа) на null.

Method	Path
GET	/secret/subkeys/:path

Параметры

- path (string: <required>) – Указывает путь секрета для чтения. Это указывается как часть URL-адреса.
- version (int: 0) - Указывает возвращаемую версию. Если не установлено, возвращается последняя версия.
- depth (int: 0) - Указывает самый глубокий уровень вложенности для предоставления в выходных данных. Значение по умолчанию 0 не накладывает никаких ограничений. Если он не равен нулю, ключи, которые находятся в указанном значении глубины, будут искусственно рассматриваться как листья и, таким образом, будут нулевыми, даже если существуют дополнительные базовые подключа.

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    https://127.0.0.1:8200/v1/secret/subkeys/my-secret?version=1
```

Пример данных секрета

```
{
  "foo": "abc",
  "bar": {
    "baz": "def"
  },
  "quux": {}
}
```

Пример ответа

```
{
  "subkeys": {
    "foo": null,
    "bar": {
      "baz": null
    },
    "quux": null
  },
  "metadata": {
    "created_time": "2021-12-14T20:28:00.773477Z",
    "custom_metadata": null,
    "deletion_time": "",
    "destroyed": false,
    "version": 1
  }
}
```

2.5.7 Удаление последней версии секрета

Endpoint выполняет обратимое удаление последней версии секрета в указанном месте. Это помечает версию как удаленную и предотвратит ее возврат при чтении, но базовые данные не будут удалены. Удаление можно отменить, используя путь восстановления.

Method	Path
`DELETE`	`/secret/data/:path`

Параметры

- path (string: <required>) – Указывает путь к удаляемому секрету. Это указывается как часть URL-адреса.

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    --request DELETE \
    https://127.0.0.1:8200/v1/secret/data/my-secret
```

2.5.8 Удаление версий секрета

Endpoint выполняет обратимое удаление указанных версий секрета. Это помечает версии как удаленные и предотвращает их возврат при чтении, но базовые данные не удаляются. Удаление можно отменить, используя путь восстановления.

Method	Path
`POST`	`/secret/delete/:path`

Параметры

- path (string: <required>) – Указывает путь к удаляемому секрету. Это указывается как часть URL-адреса.
- versions ([]int: <required>) - Версии для удаления. Версионные данные не будут удалены, но они больше не будут возвращаться в обычных запросах на получение.

Пример

```
{
  "versions": [1, 2]
}
```

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    https://127.0.0.1:8200/v1/secret/undelete/my-secret
```

2.5.9 Уничтожение версии секрета

Безвозвратно удаляет указанные данные версии для предоставленного ключа и номеров версий из хранилища 'ключ-значение'.

Method	Path
`POST`	`/secret/destroy/:path`

Параметры

- path (string: <required>) – Указывает путь секрета для уничтожения. Это указывается как часть URL-адреса.
- versions ([]int: <required>) - Версии для уничтожения. Их данные будут безвозвратно удалены.

Пример

```
{
  "versions": [1, 2]
}
```

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    https://127.0.0.1:8200/v1/secret/destroy/my-secret
```

2.5.10 Список секретов

Endpoint возвращает список имен ключей в указанном месте. Папки имеют суффикс /. Вход должен быть папкой; list в файле не вернет значение. Обратите внимание, что для ключей не выполняется фильтрация на основе политик; не кодируйте конфиденциальную информацию в именах ключей. Сами значения недоступны с помощью этой команды.

Method	Path
`LIST`	`/secret/metadata/:path`

Parameters

- path (string: <required>) – Указывает путь секретов для списка. Это указывается как часть URL-адреса.

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    --request LIST \
    https://127.0.0.1:8200/v1/secret/metadata/my-secret
```

Пример ответа

В приведенном ниже примере показаны выходные данные для пути запроса secret/, когда есть секреты в secret/foo и secret/foo/bar; обратите внимание на разницу в двух записях.

```
{
  "data": {
    "keys": ["foo", "foo/"]
  }
}
```

2.5.11 Чтение метаданных секрета

Endpoint извлекает метаданные и версии секрета по указанному пути. Метаданные не зависят от версии.

Method	Path
`GET`	`/secret/metadata/:path`

Параметры

- path (string: <required>) – Указывает путь секрета для чтения. Это указывается как часть URL-адреса.

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    https://127.0.0.1:8200/v1/secret/metadata/my-secret
```

Пример ответа

```
{
  "data": {
    "cas_required": false,
    "created_time": "2018-03-22T02:24:06.945319214Z",
    "current_version": 3,
    "delete_version_after": "3h25m19s",
    "max_versions": 0,
    "oldest_version": 0,
    "updated_time": "2018-03-22T02:36:43.986212308Z",
    "custom_metadata": {
      "foo": "abc",
      "bar": "123",
      "baz": "5c07d823-3810-48f6-a147-4c06b5219e84"
    },
    "versions": {
      "1": {
        "created_time": "2018-03-22T02:24:06.945319214Z",
        "deletion_time": "",
        "destroyed": false
      },
      "2": {
        "created_time": "2018-03-22T02:36:33.954880664Z",
        "deletion_time": "",
        "destroyed": false
      },
      "3": {
        "created_time": "2018-03-22T02:36:43.986212308Z",
        "deletion_time": "",
        "destroyed": false
      }
    }
  }
}
```


2.5.12 Создание/обновление метаданных

Endpoint создает или обновляет метаданные секрета в указанном месте. Он не создает новую версию.

Method	Path
`POST`	`/secret/metadata/:path`

Параметры

- max_versions (int: 0) – Количество версий для каждого ключа. Если не установлено, используется настроенная максимальная версия бэкэнда. Как только у ключа будет больше настроенных разрешенных версий, самая старая версия будет безвозвратно удалена.
- cas_required (bool: false) – Если значение равно true, для ключа потребуется установить параметр cas во всех запросах на запись. Если false, будет использоваться конфигурация бэкенда.
- delete_version_after (string:"0s") – Задайте для параметра delete_version_after продолжительность, чтобы указать deletion_time для всех новых версий, записываемых в этот ключ. Если этот параметр не задан, будет использоваться бэкэнд-версия_after_delete_version. Если значение больше, чем delete_version_after бэкэнда, будет использоваться бэкэнд-делет_версия_after.
- custom_metadata (map<string|string>: nil) - Сопоставление произвольной строки со строковыми метаданными, предоставленными пользователем, предназначенное для описания секрета.

Пример

```
{
  "max_versions": 5,
  "cas_required": false,
  "delete_version_after": "3h25m19s",
  "custom_metadata": {
    "foo": "abc",
    "bar": "123",
    "baz": "5c07d823-3810-48f6-a147-4c06b5219e84"
  }
}
```

Пример запроса

В приведенном ниже примере показаны выходные данные для пути запроса secret/, когда есть секреты в secret/foo и secret/foo/bar; обратите внимание на разницу в двух записях.

```
curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    https://127.0.0.1:8200/v1/secret/metadata/my-secret
```

2.5.13 Патч метаданных

Endpoint исправляет существующую запись метаданных секрета в указанном месте. Вызывающий токен должен иметь политику ACL, предоставляющую возможность исправления. В настоящее время поддерживается только исправление слияния JSON, которое должно быть указано с использованием значения заголовка Content-Type application/merge-patch+json. Он не создает новую версию.

Method	Path
`PATCH`	`/secret/metadata/:path`

Параметры

- max_versions (int: 0) – Количество версий для каждого ключа. Если не установлено, используется настроенная максимальная версия бэкэнда. Как только у ключа будет больше настроенных разрешенных версий, самая старая версия будет безвозвратно удалена.
- cas_required (bool: false) – Если значение равно true, для ключа потребуется установить параметр cas во всех запросах на запись. Если false, будет использоваться конфигурация бэкенда.
- delete_version_after (string:"0s") – Задайте для параметра delete_version_after продолжительность, чтобы указать deletion_time для всех новых версий, записываемых в этот ключ. Если этот параметр не задан, будет использоваться backend's delete_version_after. Если значение больше, чем backend's delete_version_after, будет использоваться delete_version_after.
- custom_metadata (map<string|string>: nil) - Сопоставление произвольной строки со строковыми метаданными, предоставленными пользователем, предназначенное для описания секрета.

Пример

```
{
  "max_versions": 5,
  "custom_metadata": {
    "bar": "123"
  }
}
```

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    --header "Content-Type: application/merge-patch+json"
    --request PATCH \
    --data @payload.json \
    https://127.0.0.1:8200/v1/secret/metadata/my-secret
```

2.5.14 Удаление метаданных и всех версий

Endpoint безвозвратно удаляет метаданные ключа и все данные версии для указанного ключа. Вся история версий будет удалена.

Method	Path
`DELETE`	`/secret/metadata/:path`

Параметры

- path (string: <required>) – Указывает путь к удаляемому секрету. Это указывается как часть URL-адреса.

Пример запроса

```
curl \
    --header "X-Vault-Token: ..." \
    --request DELETE \
    https://127.0.0.1:8200/v1/secret/metadata/my-secret
```