Механизм секретов Kubernetes

Этот механизм может использовать внешние сертификаты X.509 в рамках TLS или для проверки подписи. Проверка подписей с использованием сертификатов X.509, использующих SHA-1, не поддерживается без обходных решений. См. FAQ по устареванию для получения дополнительной информации.

Механизм секретов Kubernetes в StarVault генерирует токены сервисных аккаунтов Kubernetes, а также, при необходимости, создает сервисные аккаунты, привязки ролей и роли. Сгенерированные токены сервисных аккаунтов имеют настраиваемое время жизни (TTL), а все созданные объекты автоматически удаляются при истечении срока аренды (lease) в StarVault.

Для каждой аренды StarVault создает токен сервисного аккаунта, привязанный к указанному сервисному аккаунту. Этот токен возвращается вызывающему.

Чтобы узнать больше о сервисных аккаунтах в Kubernetes, ознакомьтесь с документацией по сервисным аккаунтам Kubernetes и документацией по RBAC в Kubernetes.

Не рекомендуется использовать токены, созданные механизмом секретов Kubernetes, для аутентификации через метод Kubernetes Auth в StarVault. Это приведет к созданию множества уникальных идентичностей в StarVault, что усложнит их управление.

1. Настройка

Механизм секретов Kubernetes необходимо предварительно настроить, прежде чем он сможет выполнять свои функции. Эти действия обычно выполняются оператором или системой управления конфигурацией.

1. По умолчанию StarVault подключается к Kubernetes, используя собственный сервисный аккаунт. Если используется стандартная Helm-диаграмма, этот сервисный аккаунт создаётся автоматически и носит имя, соответствующее релизу Helm (обычно starvault, но его можно изменить с помощью значения Helm-параметра `server.serviceAccount.name`).

   Необходимо убедиться, что сервисный аккаунт, который использует StarVault, имеет права для управления токенами сервисных аккаунтов, а также (опционально) для управления самими сервисными аккаунтами, ролями и привязками ролей. Эти разрешения можно задать через Kubernetes Role или ClusterRole. Роль связывается с сервисным аккаунтом StarVault с помощью RoleBinding или ClusterRoleBinding.

Пример минимальной ClusterRole для создания токенов сервисных аккаунтов:

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: k8s-minimal-secrets-abilities
rules:
- apiGroups: [""]
  resources: ["serviceaccounts/token"]
  verbs: ["create"]
```

Также можно создать более расширенную роль с полными правами на управление токенами, аккаунтами, привязками и ролями:

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: k8s-full-secrets-abilities
rules:
- apiGroups: [""]
  resources: ["serviceaccounts", "serviceaccounts/token"]
  verbs: ["create", "update", "delete"]
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["rolebindings", "clusterrolebindings"]
  verbs: ["create", "update", "delete"]
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["roles", "clusterroles"]
  verbs: ["bind", "escalate", "create", "update", "delete"]
```

Создайте эту роль в Kubernetes, например, с помощью `kubectl apply -f`.

Если вы хотите использовать выборку по меткам (label selection) для указания пространств имён, в которых роль будет действовать, StarVault должен иметь право читать объекты Namespace:

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: k8s-full-secrets-abilities-with-labels
rules:
- apiGroups: [""]
  resources: ["namespaces"]
  verbs: ["get"]
- apiGroups: [""]
  resources: ["serviceaccounts", "serviceaccounts/token"]
  verbs: ["create", "update", "delete"]
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["rolebindings", "clusterrolebindings"]
```

```
  verbs: ["create", "update", "delete"]
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["roles", "clusterroles"]
  verbs: ["bind", "escalate", "create", "update", "delete"]
```

Получение корректных прав для StarVault может потребовать проб и ошибок, поскольку Kubernetes строго ограничивает возможность повышения привилегий. Подробнее читайте в документации по RBAC Kubernetes.

Защищайте сервисный аккаунт StarVault, особенно если он имеет расширенные права — фактически он выполняет роль администратора кластера.

2. Создайте привязку (binding), связывающую эту роль с сервисным аккаунтом StarVault и предоставляющую ему право управления токенами:

YAML

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: starvault-token-creator-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: k8s-minimal-secrets-abilities
subjects:
- kind: ServiceAccount
  name: starvault
  namespace: starvault
```

Для получения дополнительной информации о ролях, сервисных аккаунтах, привязках и токенах ознакомьтесь с документацией по RBAC Kubernetes.

3. Если StarVault не будет автоматически управлять ролями или сервисными аккаунтами (см. раздел «Автоматическое управление ролями и аккаунтами»), необходимо создать сервисный аккаунт, для которого StarVault будет выдавать токены.

Крайне не рекомендуется использовать один и тот же сервисный аккаунт и для работы StarVault, и для генерации токенов.

В примерах ниже будет использоваться пространство имён `test`. Создайте его, если это пространство имён ещё не создано:

BASH

```bash
$ kubectl create namespace test
namespace/test created
```

Простой пример сервисного аккаунта, роли и привязки роли в пространстве имён `test`:

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: test-service-account-with-generated-token
  namespace: test
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: test-role-list-pods
  namespace: test
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: test-role-abilities
  namespace: test
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: test-role-list-pods
subjects:
- kind: ServiceAccount
  name: test-service-account-with-generated-token
  namespace: test
```

Создайте эти объекты с помощью `kubectl apply -f`.

4. Включите механизм секретов Kubernetes:

```bash
$ starvault secrets enable kubernetes
Success! Enabled the kubernetes Secrets Engine at: kubernetes/
```

По умолчанию механизм монтируется по имени `kubernetes/`. Это можно изменить с помощью аргумента `-path`.

5. Настройте точки монтирования. Пустая конфигурация допустима:

```bash
$ starvault write -f kubernetes/config
```

Доступные параметры конфигурации описаны в документации API.

6. Теперь вы можете настроить механизм Kubernetes Secrets Engine для создания роли StarVault (не путать с Kubernetes Role), которая может генерировать токены для заданного сервисного аккаунта:

```bash
$ starvault write kubernetes/roles/my-role \
    allowed_kubernetes_namespaces="*" \
    service_account_name="test-service-account-with-generated-token" \
    token_default_ttl="10m"
```

2. Генерация учетных данных

После того как пользователь прошёл аутентификацию в StarVault и имеет соответствующие разрешения, выполнение записи в конечную точку `creds` для заданной роли StarVault приведёт к генерации и возврату нового токена сервисного аккаунта.

+

```bash
$ starvault write kubernetes/creds/my-role \
    kubernetes_namespace=test
```

Вывод:

Key	Value
lease_id	kubernetes/creds/my-role/31d771a6-…
lease_duration	10m0s
lease_renwable	false
service_account_name	test-service-account-with-generated-token
service_account_namespace	test
service_account_token	eyJHbGci0iJSUzI1NiIsImtpZCI6ImlrUEE…

Вы можете использовать выданный токен сервисного аккаунта (`eyJHbG…`) для любых запросов к API Kubernetes, на которые у этого аккаунта есть права (через привязки ролей).

Пример запроса к API:

```bash
$ curl -sk $(kubectl config view --minify -o 'jsonpath=
{.clusters[].cluster.server}')/api/v1/namespaces/test/pods \
    --header "Authorization: Bearer eyJHbGci0iJSUzI1Ni..."
```

Пример ответа:

```json
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "resourceVersion": "1624"
  },
  "items": []
}
```

Когда срок аренды истечёт, вы можете убедиться, что токен отозван:

```bash
$ curl -sk $(kubectl config view --minify -o 'jsonpath=
{.clusters[].cluster.server}')/api/v1/namespaces/test/pods \
    --header "Authorization: Bearer eyJHbGci0iJSUzI1Ni..."
```

Пример ответа при истекшем токене:

```json
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {},
  "status": "Failure",
  "message": "Unauthorized",
  "reason": "Unauthorized",
  "code": 401
}
```

3. TTL (время жизни токена)

Токены сервисных аккаунтов Kubernetes имеют время жизни (TTL). Когда токен истекает, он автоматически отзывается.

Вы можете задать TTL по умолчанию (`token_default_ttl`) и максимальное TTL (`token_max_ttl`) при создании или настройке роли StarVault:

```bash
$ starvault write kubernetes/roles/my-role \
    allowed_kubernetes_namespaces="*" \
    service_account_name="test-service-account-with-generated-token" \
    token_default_ttl="10m" \
    token_max_ttl="2h"
```

Также можно указать **время жизни токена** (TTL) при генерации токена через конечную точку `creds` . Если значение `ttl` не указано, используется значение по умолчанию. Оно не может превышать максимальное TTL, заданное для роли:

```bash
$ starvault write kubernetes/creds/my-role \
    kubernetes_namespace=test \
    ttl=20m
```

Вывод:

Key	Value
lease_id	kubernetes/creds/my-role/31d771a6-…
lease_duration	20m0s
lease_renewable	false
service_account_name	test-service-account-with-generated-token
service_account_namespace	test
service_account_token	eyJHbGci0iJSUzI1NiIsImtpZCI6ImlrUEE…

Вы можете проверить TTL токена, расшифровав JWT и извлекая поля `iat` (время выдачи) и `exp` (время истечения):

```bash
$ echo 'eyJhbGc...' | cut -d'.' -f2 | base64 -d | jq -r '.iat,.exp|todate'
2022-05-20T17:14:50Z
2022-05-20T17:34:50Z
```

4. Аудитории

Токены сервисных аккаунтов Kubernetes поддерживают аудитории.

Вы можете задать **аудитории по умолчанию** (`token_default_audiences`) при создании или настройке роли StarVault. Если не указано явно, будут использоваться аудитории по умолчанию, заданные в настройках кластера Kubernetes.

```bash
$ starvault write kubernetes/roles/my-role \
    allowed_kubernetes_namespaces="*" \
    service_account_name="test-service-account-with-generated-token" \
    token_default_audiences="custom-audience"
```

Также можно указать аудитории (`audiences`) напрямую при генерации токена через конечную точку `creds` . Если параметр не задан, будут использоваться аудитории по умолчанию, заданные в роли:

```bash
$ starvault write kubernetes/creds/my-role \
    kubernetes_namespace=test \
    audiences="another-custom-audience"
```

Вывод:

Key	Value
lease_id	kubernetes/creds/my-role/SriWQf0bPZ…
lease_duration	768h
lease_renwable	false
service_account_name	test-service-account-with-generated-token
service_account_namespace	test
service_account_token	eyJHbGci0iJSUzI1NiIsImtpZCI6ImlrUEE…

Вы можете проверить аудитории токена, расшифровав JWT:

```bash
$ echo 'eyJhbGc...' | cut -d'.' -f2 | base64 -d
{"aud":
["another-custom-audience"
]...}
```

5. Автоматическое управление ролями и сервисными аккаунтами

При настройке роли StarVault вы можете указать параметры, позволяющие автоматически создавать сервисный аккаунт Kubernetes и привязку роли (role binding), а при необходимости — и саму роль Kubernetes.

Если вы хотите использовать **существующую роль Kubernetes**, но при этом **автоматически создавать сервисный аккаунт** и **привязку роли**, задайте параметр `kubernetes_role_name`:

```bash
$ starvault write kubernetes/roles/auto-managed-sa-role \
    allowed_kubernetes_namespaces="test" \
    kubernetes_role_name="test-role-list-pods"
```

Сервисный аккаунт StarVault также должен иметь доступ к ресурсам, к которым он предоставляет доступ. Это можно обеспечить, выполнив следующую команду

```bash
kubectl -n test create clusterrolebinding --clusterrole k8s-full-secrets-abilities
--serviceaccount=starvault:starvault starvault-test-role-abilities
```

Это часть механизма Kubernetes по защите от повышения привилегий. Подробнее — в документации по RBAC Kubernetes.

Вы можете получить учётные данные с автоматически созданным сервисным аккаунтом:

```bash
$ starvault write kubernetes/creds/auto-managed-sa-role \
    kubernetes_namespace=test
```

Вывод:

Key	Value
lease_id	kubernetes/creds/auto-managed-sa-role/cujRLYjKZUMQk6dkHBGGWm67
lease_duration	768h
lease_renwable	false
service_account_name	v-token-auto-man-1653001548-5z6hrgsxnmzncxejztml4arz
service_account_namespace	test
service_account_token	eyJHbGci0iJSUzI1Ni…

StarVault также может автоматически создавать роль, помимо сервисного аккаунта и привязки роли. Для этого используется параметр `generated_role_rules`, принимающий правила в формате JSON или YAML:

```bash
$ starvault write kubernetes/roles/auto-managed-sa-and-role \
    allowed_kubernetes_namespaces="test" \
    generated_role_rules='{"rules":[{"apiGroups":[""],"resources":
["pods"],"verbs":["list"]}]}'
```

Получение учётных данных выполняется аналогично:

```bash
$ starvault write kubernetes/creds/auto-managed-sa-and-role \
```

```
    kubernetes_namespace=test
```

Вывод:

Key	Value
lease_id	kubernetes/creds/auto-managed-sa-and-role/pehLtegoTP8vCkcaQozUqOHf
lease_duration	768h
lease_renwable	false
service_account_name	v-token-auto-man-1653002096-4imxf3ytjh5hbyro9s1oqdo3
service_account_namespace	test
service_account_token	eyJHbGci0iJSUzI1Ni…

6. API

Механизм секретов Kubernetes в StarVault предоставляет полный HTTP API. Подробнее см. в документации по Kubernetes Secrets Engine API.

Механизм секретов KV

Механизм секретов KV (сокращение от "Key-Value") — это универсальное хранилище значений ключей, используемое для хранения произвольных секретов в настроенном физическом хранилище для StarVault. Этот механизм может работать в одном из двух режимов:

- **KV версии 1 (KV v1)** - не поддерживает версионирование значений ключа. Хранит одно значения для ключа
- **KV версии 2 (KV v2)** - поддерживает версионирование значений ключа. Хранит настраиваемое количество версий для каждого ключа.

1. KV версии 1

Механизм секретов **kv** используется для хранения произвольных секретов в настроенном физическом хранилище для StarVault.

При записи в ключ в механизме **kv** старое значение будет заменено; подполя не объединяются.

Имена ключей всегда должны быть строками. Если вы напрямую записываете значения, не являющиеся строками, через интерфейс командной строки (CLI), они будут преобразованы в строки. Однако вы можете сохранить значения, не являющиеся строками, записав пары ключ/значение в StarVault из JSON-файла или используя HTTP API.

Этот механизм учитывает различие между операциями `create` и `update` в политиках контроля доступа (ACL).

Пути и имена ключей не скрываются и не шифруются; шифруются только значения, установленные для ключей. Настоятельно не рекомендуется хранить конфиденциальную информацию в части пути секрета.

1.1. Активация

▶ UI

▶ CLI

1.2. Использование

▶ UI

▶ CLI

2. KV версии 2

Механизм секретов **kv** используется для хранения произвольных секретов в настроенном физическом хранилище для StarVault.

При записи в ключ в механизме **kv** старое значение будет заменено; подполя не объединяются.

Имена ключей всегда должны быть строками. Если вы напрямую записываете значения, не являющиеся строками, через интерфейс командной строки (CLI), они будут преобразованы в строки. Однако вы можете сохранить значения, не являющиеся строками, записав пары ключ/значение в StarVault из JSON-файла или используя HTTP API.

Этот механизм учитывает различие между операциями `create` и `update` в политиках контроля доступа (ACL).

KV версии 2 может сохранять настраиваемое количество версий секретов. Это позволяет восстановить данные старых версий в случае нежелательного удаления или обновления данных. Кроме того, операции **Check-and-Set** могут использоваться для защиты данных от непреднамеренной перезаписи.

2.1. Активация

▶ UI

▶ CLI

2.2. Обновление с v1 до v2

Существующий механизм KV версии 1 можно обновить до KV версии 2. При этом начнется процесс обновления существующих данных ключей/значений до версионного формата. Во время этого процесса хранилище будет недоступно. Этот процесс может занять много времени, поэтому планируйте его заранее.

После обновления до версии 2 прежние пути, по которым можно было получить доступ к данным, больше не будут достаточными. Вам нужно будет настроить политики пользователей, чтобы добавить доступ к путям версии 2, как описано в разделе Правила ACL ниже. Аналогично, пользователям/приложениям необходимо будет обновить пути, по которым они взаимодействуют с данными KV после обновления до версии 2.

Для обновления KV с версии 1 до версии 2 можно использовать следующие способы:

Способ 1

```
starvault kv enable-versioning <path> ①
```

① <path> - путь к механизму KV версии 1

Например:

```
starvault kv enable-versioning kv/
```

Способ 2

```
starvault secrets tune -version=2 <path> ①
```

① <path> - путь к механизму KV версии 1

Например:

```
starvault secrets tune -version=2 kv/
```

2.3. Правила ACL

Подробнее о политиках и правилах доступа можно прочитать в разделе Политики.

Для механизма секретов KV версии 2 (kv-v2), пути для записи и чтения версий секретов имеют префикс **data/**. Это отличается от версии 1 (kv-v1), где такой префикс не использовался. Поэтому, если у вас есть политика, которая работала для kv-v1, вам необходимо обновить её, чтобы она соответствовала структуре путей kv-v2.

Например, следующая политика для kv-v1:

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

Должна быть изменена для kv-v2 на следующую:

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

① Обратите внимание на наличие префикса **data/**

Далее приведены примеры политик, разрешающих различные операции с секретом:

Пример 7. Право на удаление последней версии ключа:

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

Пример 8. Право на удаление любой версии ключа:

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

Пример 9. Право на отмену удаления данных:

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

Пример 10. Право на уничтожение версии:

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

Пример 11. Право на просмотр списка ключей:

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

Пример 12. Право на просмотр метаданных для каждой версии:

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

```
}
```

Пример 13. Право на окончательное удаление всех версий и метаданных для ключа:

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

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

2.4. Использование

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

▶ UI

▶ CLI

Механизм секретов LDAP

Этот механизм может использовать внешние сертификаты X.509 в рамках TLS или для проверки подписи. Проверка подписей с использованием сертификатов X.509, использующих SHA-1, не поддерживается без обходных решений. См. FAQ по устареванию для получения дополнительной информации.

Механизм секретов LDAP предоставляет возможность управлять LDAP-учетными данными, а также динамически создавать учетные записи. Поддерживаются реализации протокола LDAP v3, включая OpenLDAP, Active Directory и IBM RACF (Resource Access Control Facility).

Основные возможности:

- Статические учетные данные
- Динамические учетные данные
- Проверка/аренда сервисных аккаунтов (Service Account Check-Out)

1. Настройка

1. Включите механизм секретов LDAP:

   ```bash
   $ starvault secrets enable ldap
   ```

   По умолчанию, механизм монтируется под именем ldap. Чтобы использовать другой путь, добавьте аргумент `-path`.

2. Настройте учетные данные, которые StarVault использует для связи с LDAP для генерации паролей:

   ```bash
   $ starvault write ldap/config \
       binddn=$USERNAME \
       bindpass=$PASSWORD \
       url=ldaps://138.91.247.105
   ```

   Рекомендуется создать отдельную учетную запись управления входом специально для StarVault.

3. Смените пароль `root`, чтобы только StarVault знал учетные данные:

```
$ starvault write -f ldap/rotate-root
```

BASH

Получить сгенерированный пароль после смены в StarVault невозможно. Рекомендуется создать отдельную учетную запись управления входом специально для StarVault.

2. Схемы

Механизм секретов LDAP поддерживает три различные схемы:

- `openldap` (default)
- `racf`
- `ad`

2.1. OpenLDAP

По умолчанию механизм секретов LDAP предполагает, что пароль для входа хранится в `userPassword`. Существует множество классов объектов, которые предоставляют `userPassword`, включая, например:

- `organization`
- `organizationalUnit`
- `organizationalRole`
- `inetOrgPerson`
- `person`
- `posixAccount`

2.2. Средство контроля доступа к ресурсам (RACF)

Для управления системой безопасности IBM Resource Access Control Facility (RACF) механизм секретов должен быть настроен на использование схемы `racf`.

Для поддержки RACF генерируемые пароли должны состоять из 8 символов или меньше. Длина пароля может быть настроена с помощью политики паролей:

```
$ starvault write ldap/config \
    binddn=$USERNAME \
    bindpass=$PASSWORD \
    url=ldaps://138.91.247.105 \
```

BASH

```
    schema=racf \
    password_policy=racf_password_policy
```

3. Активная директория (Active Directory - AD)

Для управления экземплярами Active Directory механизм секретов должен быть настроен на использование схемы ad.

```bash
$ starvault write ldap/config \
    binddn=$USERNAME \
    bindpass=$PASSWORD \
    url=ldaps://138.91.247.105 \
    schema=ad
```

4. Статические учетные данные

4.1. Настройка

1. Настройте статическую роль, которая сопоставляет имя в StarVault с записью в LDAP. Настройки ротации паролей будут управляться этой ролью.

   ```bash
   $ starvault write ldap/static-role/domain \
       dn='uid=domain,ou=users,dc=domain,dc=com' \
       username='domain' \
       rotation_period="24h"
   ```

2. Запрос учетных данных для роли "domain":

   ```bash
   $ starvault read ldap/static-cred/domain
   ```

4.2. Смена пароля

Управление паролями может осуществляться двумя способами:

- автоматическая смена по времени;
- ручное изменение.

4.3. Автоматическое изменение пароля

Пароли будут автоматически изменяться в соответствии с периодом ротации (`rotation_period`), настроенным в статической роли (минимум 5 секунд). При запросе учетных данных для статической роли в ответе будет указано время до следующей ротации (`ttl`).

В настоящее время автоизменение поддерживается только для статических ролей. Учетная запись `binddn`, используемая StarVault, должна быть изменена с помощью конечной точки `rotate-root`, чтобы сгенерировать пароль, который будет известен только StarVault.

4.4. Изменение вручную

Статические роли могут быть вручную изменены с помощью конечной точки `rotate-role`. При ручном изменении период изменения начинается заново.

4.5. Удаление статических ролей

При удалении статической роли пароли не изменяются. Пароль должен быть изменен вручную перед удалением роли или отзывом доступа к статической роли.

5. Динамические учетные записи

5.1. Настройка

Динамические учетные данные можно настроить, вызвав конечную точку `/role/:role_name`:

BASH

```
$ starvault write ldap/role/dynamic-role \
  creation_ldif=@/path/to/creation.ldif \
  deletion_ldif=@/path/to/deletion.ldif \
  rollback_ldif=@/path/to/rollback.ldif \
  default_ttl=1h \
  max_ttl=24h
```

Аргумент `rollback_ldif` необязателен, но рекомендуется. Операторы, содержащиеся в `rollback_ldif`, будут выполнены, если создание по какой-либо причине завершится неудачей. Это гарантирует, что все сущности будут удалены в случае неудачи.

Для создания учетной записи выполните команду:

BASH

```
$ starvault read ldap/creds/dynamic-role
```

Вывод:

Key	Value
lease_id	ldap/creds/dynamic-role/HFgd6uKaDomVMvJpYbn9q4q5
lease_duration	1h
lease_renewable	true
distinguished_names	[cn=v_token_dynamic-role_FfH2i1c4dO_1611952635,ou=users,dc=learn,dc=example]
password	xWMjkIFMerYttEbzfnBVZvhRQGmhpAA0yeTya8fdmD-B3LXDzGrjNEPV2bCPE9CW6
username	v_token_testrole_FfH2i1c4dO_1611952635

Поле `distinguished_names` представляет собой массив DN, которые создаются из операторов `create_ldif`. Если включено более одной записи LDIF, в это поле будут включены DN из каждого утверждения. Каждая запись в этом поле соответствует одному LDIF-выражению. Дедупликации не происходит, и порядок сохраняется.

5.2. Сущности LDIF

Управление учетными записями пользователей осуществляется с помощью записей LDIF. Записи LDIF могут представлять собой base64-кодированную версию строки LDIF. Строка будет разобрана и проверена на соответствие синтаксису LDIF. Хороший справочник по правильному синтаксису LDIF можно найти здесь.

Некоторые важные моменты, которые следует помнить при создании записей LDIF:

- Ни в одной строке не должно быть пробелов, включая пустые строки;
- Каждый блок модификации должен предваряться пустой строкой;
- В одном блоке модификации можно задать несколько модификаций для `dn`. Каждая модификация должна завершаться одним тире `(-)`

5.3. Активная директория (AD)

Для AD есть несколько дополнительных деталей, которые важно помнить:

Чтобы программно создать пользователя в AD, вы сначала добавляете объект пользователя, а затем изменяете его, чтобы указать пароль и включить учетную запись.

- Пароли в AD задаются с помощью поля `unicodePwd`. Перед ним должны стоять два (2) двоеточия `(::).`
- При программной установке пароля в AD должны быть соблюдены следующие критерии:
  - Пароль должен быть заключен в двойные кавычки `(" ")`;
  - Пароль должен быть в формате **UTF16LE**;
  - Пароль должен быть в `base64`-кодировке;
  - Дополнительные сведения можно найти здесь.
- После того как пароль пользователя установлен, его можно включить. Для этого в AD используется поле `userAccountControl`:
  - Чтобы включить учетную запись, установите значение `userAccountControl` равным `512`;
  - Скорее всего, вы также захотите отключить истечение срока действия пароля AD для этой динамической учетной записи пользователя. Значение `userAccountControl` для этого следующее: 65536;
  - Флаги `userAccountControl` являются кумулятивными, поэтому, чтобы установить оба вышеуказанных флага, сложите два значения (512 + 65536 = 66048): переведите `userAccountControl` в `66048`;
  - Подробнее о флагах `userAccountControl` см. здесь.

`sAMAccountName` - это общее поле при работе с пользователями AD. Оно используется для обеспечения совместимости с устаревшими системами Windows NT и имеет ограничение в 20 символов. Помните об этом при определении шаблона `username_template`. Дополнительные сведения см. здесь.

Поскольку шаблон `username_template` по умолчанию длиннее 20 символов и соответствует шаблону `v_{{.DisplayName}}_{{.RoleName}}_{{random 10}}_{{unix_time}}`, мы рекомендуем настроить шаблон `username_template` в конфигурации роли, чтобы генерировать учетные записи с именами менее 20 символов. Для получения дополнительной информации обратитесь к документу о шаблонах имен пользователей.

Что касается добавления динамических пользователей в группы, AD не позволяет напрямую изменять атрибут `memberOf` пользователя. Атрибут `member` группы и атрибут `memberOf` пользователя являются связанными атрибутами. Связанные атрибуты представляют собой пары прямая ссылка/обратная ссылка, причем прямая ссылка может быть изменена. В случае членства в группе AD атрибут `member` группы является прямой ссылкой. Чтобы добавить вновь созданного динамического пользователя в группу, нам также нужно отправить запрос `modify` в нужную группу и обновить членство в группе для нового пользователя.

5.3.1. Пример LDIF для Active Directory

Различные параметры `*_ldif` представляют собой шаблоны, использующие язык шаблонов **go**. Полный пример LDIF для создания учетной записи пользователя Active Directory приведен здесь для справки:

```ldif
dn: CN={{.Username}},OU=domain,DC=adtesting,DC=lab
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
userPrincipalName: {{.Username}}@adtesting.lab
sAMAccountName: {{.Username}}

dn: CN={{.Username}},OU=domain,DC=adtesting,DC=lab
changetype: modify
replace: unicodePwd
unicodePwd::{{ printf "%q" .Password | utf16le | base64 }}
-
replace: userAccountControl
userAccountControl: 66048
-

dn: CN=test-group,OU=domain,DC=adtesting,DC=lab
changetype: modify
add: member
member: CN={{.Username}},OU=domain,DC=adtesting,DC=lab
-
```

6. Проверка учетной записи сервиса

Проверка учетной записи сервиса предоставляет библиотеку учетных записей сервиса, которые могут быть проверены человеком или машиной. StarVault автоматически меняет пароль при каждой регистрации учетной записи. Учетные записи сервисов можно регистрировать добровольно, или StarVault будет регистрировать их, когда закончится период кредитования ("ttl").

Функциональность проверки учетных записей сервисов работает с различными схемами, включая OpenLDAP, Active Directory и RACF. В следующем примере использования механизм секретов настроен на управление библиотекой учетных записей сервисов в экземпляре Active Directory.

Сначала нам нужно включить механизм секретов LDAP и указать ему, как безопасно подключиться к серверу AD.

```bash
$ starvault secrets enable ldap
Success! Enabled the ad secrets engine at: ldap/

$ starvault write ldap/config \
    binddn=$USERNAME \
    bindpass=$PASSWORD \
    url=ldaps://138.91.247.105 \
    userdn='dc=example,dc=com'
```

Следующим шагом будет назначение набора учетных записей сервисов для проверки.

```bash
$ starvault write ldap/library/accounting-team \
    service_account_names=fizz@example.com,buzz@example.com \
    ttl=10h \
    max_ttl=20h \
    disable_check_in_enforcement=false
```

В этом примере имена учетных записей сервисов `fizz@example.com` и `buzz@example.com` уже созданы на удаленном сервере AD. Они были выделены исключительно для работы StarVault.

`ttl` - это время, в течение которого каждая проверка будет длиться до того, как StarVault проверит учетную запись сервиса, сменив ее пароль во время регистрации.

`max_ttl` - максимальное количество времени, которое она может прожить, если ее обновить.

По умолчанию они равны 24 часам, и в обоих случаях используются строки формата duration. Также по умолчанию учетная запись сервиса должна регистрироваться той же сущностью StarVault или клиентским токеном, которая ее зарегистрировала. Однако если такое поведение вызывает проблемы, установите `disable_check_in_enforcement=true`.

После создания библиотеки учетных записей сервисов вы можете в любой момент просмотреть их статус, чтобы узнать, доступны ли они или проверены.

```bash
$ starvault read ldap/library/accounting-team/status
```

Вывод:

Key	Value
buzz@example.com	map[available:true]
fizz@example.com	map[available:true]

Чтобы проверить любую доступную учетную запись сервисов, просто выполните команду:

```
$ starvault write -f ldap/library/accounting-team/check-out
```

BASH

Вывод:

Key	Value
lease_id	ldap/library/accounting-team/check-out/EpuS8cX7uEsDzOwW9kkKOyGW
lease_duration	10h
lease_renewable	true
password	?@09AZKh03hBORZPJcTDgLfntlHqxLy29tcQjPVThzuwWAx/Twx4a2Zc-RQRqrZ1w
service_account_name	fizz@example.com

Если значение `ttl` по умолчанию выше, чем нужно, установите более короткое время с помощью:

```
$ starvault write ldap/library/accounting-team/check-out ttl=30m
```

BASH

Вывод:

Key	Value
lease_id	ldap/library/accounting-team/check-out/gMonJ2jB6kYs6d3Vw37WFDCY
lease_duration	30m
lease_renewable	true
password	?@09AZerLLuJfEMbRqP+3yfQYDSq6laP48TCJRBJaJu/kDKLsq9WxL9szVAvL/E1
service_account_name	buzz@example.com

Это может быть хорошим способом сказать: "Хотя я могу выписать учетную запись на 24 часа, если я не зарегистрировал ее через 30 минут, значит, я забыл или я экземпляр с

истекшим сроком использования, поэтому вы можете просто зарегистрировать учетную запись обратно".

Если ни одна учетная запись службы не доступна для выгрузки, StarVault вернет `400 Bad Request`.

```
$ starvault write -f ldap/library/accounting-team/check-out
Error writing data to ldap/library/accounting-team/check-out: Error making API
request.

URL: POST http://localhost:8200/v1/ldap/library/accounting-team/check-out
Code: 400. Errors:

* No service accounts available for check-out.
```

Чтобы продлить работу учетной записи, возобновите ее аренду.

```
$ starvault lease renew ldap/library/accounting-team/check-
out/0C2wmeaDmsToVFc0zDiX9cMq
```

Вывод:

Key	Value
lease_id	ldap/library/accounting-team/check-out/0C2wmeaDmsToVFc0zDiX9cMq
lease_duration	10h
lease_renewable	true

Продление регистрации означает, что текущий пароль будет жить дольше, так как пароли меняются каждый раз, когда пароль проверяется либо вызовом, либо StarVault, потому что срок проверки заканчивается.

Чтобы снова зарегистрировать служебную учетную запись для использования другими пользователями, вызовите:

```
$ starvault write -f ldap/library/accounting-team/check-in
```

Вывод:

Key	Value

| --- | ---- |
| check_ins | [fizz@example.com] |

В большинстве случаев это срабатывает, но если один и тот же абонент проверяет несколько аккаунтов, StarVault нужно будет знать, какой из них регистрировать.

```
$ starvault write ldap/library/accounting-team/check-in
service_account_names=fizz@example.com
```

Вывод:

Key	Value
check_ins	[fizz@example.com]

Чтобы выполнить регистрацию, StarVault проверяет, может ли пользователь зарегистрировать данную учетную запись службы. Для этого StarVault ищет либо тот же идентификатор сущности, который использовался для проверки учетной записи службы, либо тот же клиентский токен.

Если пользователь не может проверить учетную запись сервиса или просто не пытается это сделать, StarVault автоматически проверит ее по истечении ttl. Однако если это слишком долго, учетные записи сервисов могут быть принудительно зарегистрированы высокопривилегированным пользователем:

```
$ starvault write -f ldap/library/manage/accounting-team/check-in
```

Вывод:

Key	Value
check_ins	[fizz@example.com]

Или, альтернативно, отзыв аренды секрета приводит к тому же результату.

```
$ starvault lease revoke ldap/library/accounting-team/check-
out/PvBVG0m7pEg2940Cb3Jw3KpJ
```

```
All revocation operations queued successfully!
```

7. Генерация паролей

Ранее этот механизм позволял настраивать длину пароля, который генерируется при ротации учетных данных. Этот способ был признан устаревшим начиная с Vault 1.5 в пользу использования политик паролей. Это означает, что поле `length` больше не следует использовать. Следующая политика паролей может быть использована для воспроизведения аналогичного поведения, которое обеспечивалось через поле `length`:

BASH

```
length=<length>
rule "charset" {
  charset = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
}
```

8. Политика паролей LDAP

Механизм секретов LDAP не хэширует и не шифрует пароли перед тем, как изменить значения в LDAP. Такое поведение может привести к тому, что пароли будут храниться в открытом виде.

Чтобы избежать хранения паролей в открытом виде, сервер LDAP должен быть настроен с использованием политики паролей LDAP (ppolicy — не путать с политикой паролей StarVault). Политика ppolicy позволяет, например, автоматически хэшировать открытые пароли.

Ниже приведён пример политики паролей LDAP, обеспечивающей хэширование паролей в дереве данных (DIT) `dc=domain,dc=com`:

LDIF

```
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: ppolicy

dn: olcOverlay={2}ppolicy,olcDatabase={1}mdb,cn=config
changetype: add
objectClass: olcPPolicyConfig
objectClass: olcOverlayConfig
olcOverlay: {2}ppolicy
olcPPolicyDefault: cn=default,ou=pwpolicies,dc=domain,dc=com
olcPPolicyForwardUpdates: FALSE
olcPPolicyHashCleartext: TRUE
olcPPolicyUseLockout: TRUE
```

9. API

Механизм секретов LDAP предоставляет полноценный HTTP API. Подробнее см. в документации по API LDAP Secrets Engine.