Методы аутентификации. LDAP Meta

Метод `ldap-meta` auth позволяет выполнять аутентификацию с использованием существующего сервера LDAP и учетных данных user/password. Это позволяет интегрировать StarVault в среду, использующую LDAP, без дублирования конфигурации user/password в нескольких местах. Также позволяет приводить в соответствие параметры учетной записи из LDAP в Entity раздела Custom metadata, что дает возможность идентифицировать пользователя не только по ID.

Сопоставление групп и пользователей в LDAP Meta с политиками StarVault помощью путей `users/` и `groups/`.

1. Примечание по экранированию

Администратор должен обеспечить правильное преобразование DN. Это включает в себя DN пользователя, bind DN для поиска и т. д.

Единственный способ экранирования DN, выполняемый этим методом, - это имена пользователей, указанные во время входа в систему, когда они вставляются в окончательный DN привязки, и использует правила экранирования, определенные в RFC 4514.

Кроме того, в Active Directory есть правила экранирования, которые немного отличаются от RFC; в частности, требуется экранировать '#' независимо от позиции в DN (RFC требует, чтобы он экранировался только тогда, когда это первый символ), и '=', который, как указывает RFC, может быть экранирован с помощью обратной косой черты '\', но не содержит в своем наборе обязательных экранирующих символов. Если вы используете Active Directory и они отображаются в ваших именах пользователей, пожалуйста, убедитесь, что они экранированы, в дополнение к тому, что они должным образом экранированы в вашем настроенном DNs.

Для справки смотрите RFC 4514 и этот пост TechNet о символах для экранирования в Active Directory.

2. Аутентификация

2.1. Через CLI

```
$ starvault login -method=ldap-meta username=mitchellh
Password (will be hidden):
Successfully authenticated! The policies that are associated
with this token are listed below:

admins
```

2.2. Через API

```
$ curl \
    --request POST \
    --data '{"password": "foo"}' \
    http://127.0.0.1:8200/v1/auth/ldap-meta/login/mitchellh
```

Ответ будет представлен в формате JSON. Например:

```
{
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": null,
  "auth": {
    "client_token": "c4f280f6-fdb2-18eb-89d3-589e2e834cdb",
    "policies": [
      "admins"
    ],
    "metadata": {
      "username": "mitchellh"
    },
    "lease_duration": 0,
    "renewable": false
  }
}
```

3. Конфигурация

Методы аутентификации должны быть настроены заранее, прежде чем пользователи или машины смогут пройти аутентификацию. Эти шаги обычно выполняются оператором или инструментом управления конфигурацией.

1. Включите метод аутентификации LDAP Meta:

   ```
   $ starvault auth enable ldap-meta
   ```

2. Настройте параметры подключения к серверу LDAP, информацию о том, как аутентифицировать пользователей, и инструкции о том, как запрашивать членство в группах. Параметры конфигурации распределены по категориям и подробно описаны ниже.

3.1. Параметры соединения

- `url` (string, обязательно) - сервер LDAP для подключения.

  Пример 1. Примеры:

  > `ldap-meta://ldap-meta.myorg.com`, `ldaps://ldap-meta.myorg.com:636`. Это также может быть список URL через запятую, например, `ldap-meta://ldap-meta.myorg.com`, `ldaps://ldap-meta.myorg.com:636`, в этом случае серверы будут опробованы в порядке очереди, если в процессе подключения возникнут ошибки.

- `starttls` (bool, необязательно) - если true, выдает команду StartTLS после установления незашифрованного соединения.
- `insecure_tls` - (bool, необязательно) - если true, пропускает проверку SSL-сертификата LDAP-сервера - небезопасно, используйте с осторожностью!
- `certificate` - (string, необязательно) - сертификат CA для использования при проверке сертификата сервера LDAP, должен быть в кодировке x509 PEM.
- `client_tls_cert` - (string, необязательно) - Сертификат клиента для предоставления серверу LDAP, должен быть закодирован x509 PEM.
- `client_tls_key` - (string, необязательно) - ключ сертификата клиента для предоставления серверу LDAP, должен быть закодирован x509 PEM.
- `user additional properties` - конфигурация для маппинга дополнительных параметров пользователя.

  Применить конфигурацию можно двумя способами:

  > ▸ **С помощью веб-интерфейса**

  > ▸ **С помощью CLI**

3.2. Параметры привязки

Существует два альтернативных метода разрешения объекта пользователя, используемого для аутентификации конечного пользователя:

1. **Search**: при использовании Search привязка может быть анонимной или аутентифицированной.
2. **User Principal Name**: метод определения пользователей, поддерживаемый Active Directory.

3.2.1. Привязка - аутентифицированный поиск

3.2.2. Привязка - аутентифицированный поиск

- `binddn` (string, необязательно) - отличительное имя объекта для привязки при выполнении поиска пользователей и групп.

  Пример: `cn=starvault,ou=Users,dc=example,dc=com`

- `bindpass` (string, необязательно) - пароль, используемый вместе с binddn при поиске пользователя.
- `userdn` (string, необязательно) - базовый DN, по которому будет выполняться поиск пользователей.

  Пример: `ou=Users,dc=example,dc=com`

- `userattr` (string, необязательно) - атрибут объекта атрибута пользователя, соответствующий имени пользователя, переданному при аутентификации.

  Примеры: `sAMAccountName`, `cn`, `uid`

- `userfilter` (string, необязательно) - шаблон Go, используемый для построения фильтра поиска пользователей ldap meta. Шаблон может обращаться к следующим контекстным переменным: [`UserAttr`, `Username`]. По умолчанию используется фильтр (`{{.UserAttr}}={{.Username}}`) или (`userPrincipalName={{.Username}}@UPNDomain`), если задан параметр `upndomain`. Фильтр поиска пользователей можно использовать для ограничения числа пользователей, которые могут попытаться войти в систему.

  Например, чтобы ограничить вход для пользователей, которые не являются подрядчиками, можно написать (`&(objectClass=user)({{.UserAttr}}={{.Username}})(!(employeeType=Contractor))`).

При указании `userfilter` в фильтре должно присутствовать либо шаблонное значение `{{.UserAttr}}`, либо буквальное значение, соответствующее `userattr`, чтобы гарантировать, что поиск возвращает уникальный результат, учитывающий `userattr` для целей сопоставления псевдонимов сущностей, и избежать возможных коллизий при входе в систему.

3.2.3. Привязка - анонимный поиск

- `discoverdn` (bool, необязательно) - если true, используйте анонимную привязку для обнаружения DN привязки пользователя.
- `userdn` (string, необязательно) - базовый DN, по которому будет выполняться поиск пользователей.

  Пример: `ou=Users,dc=example,dc=com`

- `userattr` (string, необязательно) - атрибут объекта атрибута пользователя, соответствующий имени пользователя, переданному при аутентификации.

  Примеры: `sAMAccountName, cn, uid`

- `userfilter` (string, необязательно) - шаблон Go, используемый для построения фильтра поиска пользователей ldap meta. Шаблон может обращаться к следующим контекстным переменным: [`UserAttr`, `Username`]. По умолчанию используется фильтр (`{{.UserAttr}}={{.Username}}`) или (`userPrincipalName={{.Username}}@UPNDomain`), если задан параметр `upndomain`. Фильтр поиска пользователей можно использовать для ограничения числа пользователей, которые могут попытаться войти в систему.

  Например, чтобы ограничить вход для пользователей, которые не являются подрядчиками, можно написать `(&(objectClass=user)({{.UserAttr}}={{.Username}})(!(employeeType=Contractor)))`.

- `deny_null_bind` (bool, необязательно) - эта опция предотвращает обход аутентификации пользователями при предоставлении пустого пароля. По умолчанию используется значение `true`.
- `anonymous_group_search` (bool, необязательно) - использует анонимные привязки при выполнении поиска групп LDAP Meta. По умолчанию имеет значение `false`.

При указании `userfilter` в фильтре должно присутствовать либо шаблонное значение `{{.UserAttr}}`, либо буквальное значение, соответствующее `userattr`, чтобы гарантировать, что поиск возвращает уникальный результат, учитывающий `userattr` для целей сопоставления псевдонимов сущностей, и избежать возможных коллизий при входе в систему.

3.2.4. Разыменование псевдонимов

- `dereference_aliases` (string, необязательно) - управление тем, как будут разыменовываться псевдонимы при выполнении поиска. Возможные значения: `never`, `finding`, `searching`, и `always`. При использовании `finding` псевдонимы будут разыменовываться только во время разрешения имен базы. при использовании searching` псевдонимы будут разыменовываться после разрешения имен.

3.2.5. Привязка - основное имя пользователя (AD)

- `upndomain` (string, необязательно) - `userPrincipalDomain`, используемый для построения строки UPN для аутентифицируемого пользователя. Сконструированный UPN будет выглядеть как `[username]@UPNDomain`.

  > Пример: `example.com`, что приведет к привязке StarVault как `username@example.com`.

3.3. Разрешение членства в группе

После аутентификации пользователя метод LDAP Meta auth должен знать, как определить, к каким группам принадлежит пользователь. Конфигурация для этого может отличаться в зависимости от вашего сервера LDAP и схемы каталогов. При определении принадлежности к группе используются две основные стратегии: первая - поиск объекта authenticated user и следование атрибуту групп, членом которых он является. Вторая - поиск объектов group, членом которых является аутентифицированный пользователь. Поддерживаются оба метода.

- `groupfilter` (string, необязательно) - шаблон Go, используемый при построении запроса на членство в группе. Шаблон может обращаться к следующим контекстным переменным: [`UserDN`, `Username`]. По умолчанию используется (`|(memberUid={{.Username}})(member={{.UserDN}})(uniqueMember={{.UserDN}})`), который совместим с несколькими распространенными схемами каталогов. Для поддержки разрешения вложенных групп в Active Directory вместо этого используйте следующий запрос: `(&(objectClass=group)(member:1.2.840.113556.1.4.1941:={{.UserDN}}))`.
- `groupdn` (string, обязательно) - база поиска LDAP, используемая для поиска членства в группе. Это может быть корень, содержащий либо группы, либо пользователей.

  > Пример: `ou=Groups,dc=example,dc=com`

- `groupdn` (string, обязательно) - база поиска LDAP, используемая для поиска членства в группе. Это может быть корень, содержащий либо группы, либо пользователей.

  > Пример: `ou=Groups,dc=example,dc=com`

При использовании аутентифицированного поиска для параметров привязки (см. выше) для поиска группы используется отличительное имя, определенное для `binddn`. В противном случае для поиска группы используется аутентифицированный пользователь.

Для получения большей информации используйте `starvault path-help`.

3.4. Другие

- `username_as_alias` (bool, необязательно) - если установлено значение `true`, это заставляет метод auth использовать имя пользователя, переданное пользователем, в качестве имени псевдонима.
- `max_page_size` (int, необязательно) - если установлено значение больше 0, бэкэнд LDAP будет использовать управление постраничным поиском сервера LDAP для запроса страниц до указанного размера. Это можно использовать, чтобы избежать столкновения с ограничением максимального размера результатов сервера LDAP. В противном случае бэкэнд LDAP не будет использовать управление постраничным поиском.

4. Примеры

4.1. Сценарий 1

- Сервер LDAP, работает на `ldap.example.com`, порт 389.
- Сервер поддерживает команду `STARTTLS` для запуска шифрования на стандартном порту.
- Сертификат CA хранится в файле с именем `ldap_ca_cert.pem`
- Сервер Active Directory поддерживает атрибут userPrincipalName. Пользователи идентифицируются как `username@example.com`.
- Группы являются вложенными, мы будем использовать `LDAP_MATCHING_RULE_IN_CHAIN` для просмотра списка.
- Поиск группы начнется в разделе `ou=Groups,dc=example,dc=com`. Для всех объектов группы, расположенных по этому пути, атрибут `member` будет проверен на соответствие аутентифицированному пользователю.
- Имена групп идентифицируются с помощью их атрибута `cn`.

BASH

```
$ starvault write auth/ldap-meta/config \
    url="ldap-meta://ldap-meta.example.com" \
    userdn="ou=Users,dc=example,dc=com" \
    groupdn="ou=Groups,dc=example,dc=com" \
    groupfilter="(&(objectClass=group)(member:1.2.840.113556.1.4.1941:=
{{.UserDN}}))" \
```

```
    groupattr="cn" \
    upndomain="example.com" \
    certificate=@ldap_meta_ca_cert.pem \
    insecure_tls=false \
    starttls=true
...
```

4.2. Сценарий 2

- Сервер LDAP, работает на `ldap.example.com`, порт 389.
- Сервер поддерживает команду `STARTTLS` для запуска шифрования на стандартном порту.
- Сертификат CA хранится в файле с именем `ldap_ca_cert.pem`
- Сервер не разрешает анонимные привязки для выполнения поиска пользователей.
- Для поиска используется учетная запись привязки `cn=starvault,ou=users,dc=example,dc=com` с паролем `My$ecrt3tP4ss`.
- Объекты пользователя находятся в разделе `ou=Users,dc=example,dc=com`.
- Имя пользователя, передаваемое в хранилище при аутентификации, сопоставляется с атрибутом `sAMAccountName`.
- Принадлежность к группе будет определяться с помощью атрибута `memberOf` объектов user. Поиск начнется в разделе `ou=Users,dc=example,dc=com`.

BASH

```
$ starvault write auth/ldap-meta/config \
    url="ldap://ldap.example.com" \
    userattr=sAMAccountName \
    userdn="ou=Users,dc=example,dc=com" \
    groupdn="ou=Users,dc=example,dc=com" \
    groupfilter="(&(objectClass=person)(uid={{.Username}}))" \
    groupattr="memberOf" \
    binddn="cn=starvault,ou=users,dc=example,dc=com" \
    bindpass='My$ecrt3tP4ss' \
    certificate=@ldap_ca_cert.pem \
    insecure_tls=false \
    starttls=true
...
```

4.3. Сценарий 3

- Сервер LDAP работает на `ldap.example.com`, порт 636 (LDAPS)
- Сертификат CA хранится в файле с именем `ldap_ca_cert.pem`
- Пользовательские объекты находятся в разделе `ou=Users,dc=example,dc=com`.

- Имя пользователя, передаваемое в хранилище при аутентификации, сопоставляется с атрибутом `uid`.
- Имя пользователя для привязки будет автоматически определено с помощью анонимной привязки.
- Принадлежность к группе будет определяться с помощью любого из атрибутов `memberUid`, `member member` или `uniqueMember`. Этот поиск начнется в разделе `ou=Groups,dc=example,dc=com`.
- Названия групп определяются с помощью атрибута `cn`.

BASH

```
$ starvault write auth/ldap-meta/config \
    url="ldaps://ldap.example.com" \
    userattr="uid" \
    userdn="ou=Users,dc=example,dc=com" \
    discoverdn=true \
    groupdn="ou=Groups,dc=example,dc=com" \
    certificate=@ldap_ca_cert.pem \
    insecure_tls=false \
    starttls=true
...
```

5. Группа LDAP Meta → разработка политики

Далее мы хотим создать сопоставление группы LDAP Meta с политикой StarVault:

BASH

```
$ starvault write auth/ldap-meta/groups/scientists policies=foo,bar
```

Это сопоставляет LDAP-группу "scientists" с политиками StarVault "foo" и "bar". Мы также можем добавить определенных пользователей LDAP Meta в дополнительные (потенциально не-LDAP) группы. Обратите внимание, что политики могут быть указаны и для пользователей LDAP.

BASH

```
$ starvault write auth/ldap-meta/groups/engineers policies=foobar
$ starvault write auth/ldap-meta/users/tesla groups=engineers policies=zoobar
```

Это добавляет пользователя LDAP "tesla" в группу "engineers", которая сопоставлена с политикой "foobar" StarVault. Сам пользователь "tesla" связан с политикой "zoobar".

Наконец, мы можем проверить это, пройдя аутентификацию:

BASH

```
$ starvault login -method=ldap-meta username=tesla
Password (will be hidden):
Successfully authenticated! The policies that are associated
with this token are listed below:
```

```
default, foobar, zoobar
```

6. Примечание по сопоставлению политик

Следует отметить, что сопоставление пользователь → политика происходит во время создания токена. И изменения в членстве группы на LDAP-сервере не повлияют на токены, которые уже были предоставлены. Чтобы увидеть эти изменения, необходимо отозвать старые токены и попросить пользователя пройти повторную аутентификацию.

7. Блокировка пользователя

Если пользователь несколько раз подряд предоставит неверные учетные данные, StarVault на некоторое время прекратит попытки проверить его учетные данные, а вместо этого сразу же выдаст ошибку об отказе в разрешении. Мы называем такое поведение "блокировкой пользователя". Время, на которое пользователь будет заблокирован, называется "длительностью блокировки". Пользователь сможет войти в систему после истечения срока блокировки. Количество неудачных попыток входа, после которых пользователь будет заблокирован, называется "порог блокировки". Счетчик порога блокировки обнуляется через несколько минут без попыток входа или при успешной попытке входа. Время, в течение которого счетчик будет обнулен после отсутствия попыток входа, называется "сброс счетчика блокировки". Это позволяет предотвратить как автоматические, так и целевые запросы, то есть атаки на угадывание 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:

- его можно отключить глобально с помощью переменной окружения `VAULT_DISABLE_USER_LOCKOUT`.
- его можно отключить для всех поддерживаемых методов аутентификации (ldap-meta, userpass и approle) или для определенного поддерживаемого метода аутентификации с помощью параметра `disable_lockout` в строке `user_lockout` в конфигурационном файле. Более подробная информация приведена в разделе Конфигурация блокировки пользователей.
- его можно отключить для конкретного монтирования с помощью команды "auth tune". Более подробную информацию можно найти в команде auth tune или auth tune api.

Эта функция поддерживается только методами userpass, ldap-meta и approle auth.

8. API

Метод LDAP Meta auth имеет полноценный HTTP API. Более подробная информация приведена в разделе API метода LDAP Meta auth.

Методы аутентификации. LDAP

Данный механизм может использовать внешние сертификаты X.509 как часть проверки TLS или подписи. Проверка подписей по сертификатам X.509, использующим SHA-1, устарела и больше не используется без обходного пути. Дополнительную информацию см. в FAQ по устареванию.

Метод `ldap` auth позволяет выполнять аутентификацию с использованием существующего сервера LDAP и учетных данных user/password. Это позволяет интегрировать StarVault в среду, использующую LDAP, без дублирования конфигурации user/password в нескольких местах.

Сопоставление групп и пользователей в LDAP с политиками StarVault помощью путей `users/` и `groups/`.

1. Примечание по экранированию

Администратор должен обеспечить правильное преобразование DN. Это включает в себя DN пользователя, bind DN для поиска и т. д.

Единственный способ экранирования DN, выполняемый этим методом, - это имена пользователей, указанные во время входа в систему, когда они вставляются в окончательный DN привязки, и использует правила экранирования, определенные в RFC 4514.

Кроме того, в Active Directory есть правила экранирования, которые немного отличаются от RFC; в частности, требуется экранировать '#' независимо от позиции в DN (RFC требует, чтобы он экранировался только тогда, когда это первый символ), и '=', который, как указывает RFC, может быть экранирован с помощью обратной косой черты '\', но не содержит в своем наборе обязательных экранирующих символов. Если вы используете Active Directory и они отображаются в ваших именах пользователей, пожалуйста, убедитесь, что они экранированы, в дополнение к тому, что они должным образом экранированы в вашем настроенном DNs.

Для справки смотрите RFC 4514 и этот пост TechNet о символах для экранирования в Active Directory.

2. Аутентификация

2.1. Через CLI

```bash
$ starvault login -method=ldap username=mitchellh
Password (will be hidden):
Successfully authenticated! The policies that are associated
with this token are listed below:

admins
```

2.2. Через API

```bash
$ curl \
    --request POST \
    --data '{"password": "foo"}' \
    http://127.0.0.1:8200/v1/auth/ldap/login/mitchellh
```

Ответ будет представлен в формате JSON. Например:

```json
{
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": null,
  "auth": {
    "client_token": "c4f280f6-fdb2-18eb-89d3-589e2e834cdb",
    "policies": [
      "admins"
    ],
    "metadata": {
      "username": "mitchellh"
    },
    "lease_duration": 0,
    "renewable": false
  }
}
```

3. Конфигурация

Методы аутентификации должны быть настроены заранее, прежде чем пользователи или машины смогут пройти аутентификацию. Эти шаги обычно выполняются оператором или инструментом управления конфигурацией.

1. Включите метод аутентификации LDAP:

   ```bash
   $ starvault auth enable ldap
   ```

2. Настройте параметры подключения к серверу LDAP, информацию о том, как аутентифицировать пользователей, и инструкции о том, как запрашивать членство в группах. Параметры конфигурации распределены по категориям и подробно описаны ниже.

3.1. Параметры соединения

- `url` (string, обязательно) - сервер LDAP для подключения.

  > Примеры: `ldap://ldap.myorg.com`, `ldaps://ldap.myorg.com:636`. Это также может быть список URL через запятую, например, `ldap://ldap.myorg.com`, `ldaps://ldap.myorg.com:636`, в этом случае серверы будут опробованы в порядке очереди, если в процессе подключения возникнут ошибки.

- `starttls` (bool, необязательно) - если true, выдает команду StartTLS после установления незашифрованного соединения.
- `insecure_tls` - (bool, необязательно) - если true, пропускает проверку SSL-сертификата LDAP-сервера - небезопасно, используйте с осторожностью!
- `certificate` - (string, необязательно) - сертификат CA для использования при проверке сертификата сервера LDAP, должен быть в кодировке x509 PEM.
- `client_tls_cert` - (string, необязательно) - Сертификат клиента для предоставления серверу LDAP, должен быть закодирован x509 PEM.
- `client_tls_key` - (string, необязательно) - ключ сертификата клиента для предоставления серверу LDAP, должен быть закодирован x509 PEM.

3.2. Параметры привязки

Существует два альтернативных метода разрешения объекта пользователя, используемого для аутентификации конечного пользователя:

1. **Search**: при использовании Search привязка может быть анонимной или аутентифицированной.
2. **User Principal Name**: метод определения пользователей, поддерживаемый Active Directory.

3.2.1. Привязка - аутентифицированный поиск

- `binddn` (string, необязательно) - отличительное имя объекта для привязки при выполнении поиска пользователей и групп.

> Пример: `cn=starvault,ou=Users,dc=example,dc=com`

- `bindpass` (string, необязательно) - пароль, используемый вместе с binddn при поиске пользователя.
- `userdn` (string, необязательно) - базовый DN, по которому будет выполняться поиск пользователей.

> Пример: `ou=Users,dc=example,dc=com`

- `userattr` (string, необязательно) - атрибут объекта атрибута пользователя, соответствующий имени пользователя, переданному при аутентификации.

> Примеры: `sAMAccountName`, `cn`, `uid`

- `userfilter` (string, необязательно) - шаблон Go, используемый для построения фильтра поиска пользователей ldap. Шаблон может обращаться к следующим контекстным переменным: [`UserAttr`, `Username`]. По умолчанию используется фильтр (`{{.UserAttr}}={{.Username}}`) или (`userPrincipalName={{.Username}}@UPNDomain`), если задан параметр `upndomain`. Фильтр поиска пользователей можно использовать для ограничения числа пользователей, которые могут попытаться войти в систему.

> Например, чтобы ограничить вход для пользователей, которые не являются подрядчиками, можно написать (`&(objectClass=user)({{.UserAttr}}={{.Username}})(!(employeeType=Contractor))`).

> При указании `userfilter` в фильтре должно присутствовать либо шаблонное значение `{{.UserAttr}}`, либо буквальное значение, соответствующее `userattr`, чтобы гарантировать, что поиск возвращает уникальный результат, учитывающий `userattr` для целей сопоставления псевдонимов сущностей, и избежать возможных коллизий при входе в систему.

3.2.2. Привязка - анонимный поиск

- `discoverdn` (bool, необязательно) - если true, используйте анонимную привязку для обнаружения DN привязки пользователя.
- `userdn` (string, необязательно) - базовый DN, по которому будет выполняться поиск пользователей.

> Пример: `ou=Users,dc=example,dc=com`

- `userattr` (string, необязательно) - атрибут объекта атрибута пользователя, соответствующий имени пользователя, переданному при аутентификации.

> Примеры: `sAMAccountName, cn, uid`

- `userfilter` (string, необязательно) - шаблон Go, используемый для построения фильтра поиска пользователей ldap. Шаблон может обращаться к следующим контекстным переменным: [`UserAttr`, `Username`]. По умолчанию используется фильтр (`{{.UserAttr}}={{.Username}}`) или (`userPrincipalName={{.Username}}@UPNDomain`), если задан параметр `upndomain`. Фильтр поиска пользователей можно использовать для ограничения числа пользователей, которые могут попытаться войти в систему.

  > Например, чтобы ограничить вход для пользователей, которые не являются подрядчиками, можно написать `(&(objectClass=user)({{.UserAttr}}={{.Username}})(!(employeeType=Contractor)))`.

- `deny_null_bind` (bool, необязательно) - эта опция предотвращает обход аутентификации пользователями при предоставлении пустого пароля. По умолчанию используется значение `true`.
- `anonymous_group_search` (bool, необязательно) - использует анонимные привязки при выполнении поиска групп LDAP. По умолчанию имеет значение `false`.

> При указании `userfilter` в фильтре должно присутствовать либо шаблонное значение `{{.UserAttr}}`, либо буквальное значение, соответствующее `userattr`, чтобы гарантировать, что поиск возвращает уникальный результат, учитывающий `userattr` для целей сопоставления псевдонимов сущностей, и избежать возможных коллизий при входе в систему.

3.2.3. Разыменование псевдонимов

- `dereference_aliases` (string, необязательно) - управление тем, как будут разыменовываться псевдонимы при выполнении поиска. Возможные значения: `never`, `finding`, `searching`, и `always`. При использовании `finding` псевдонимы будут разыменовываться только во время разрешения имен базы. при использовании searching` псевдонимы будут разыменовываться после разрешения имен.

3.2.4. Привязка - основное имя пользователя (AD)

- `upndomain` (string, необязательно) - `userPrincipalDomain`, используемый для построения строки UPN для аутентифицируемого пользователя. Сконструированный UPN будет выглядеть как `[username]@UPNDomain`.

  > Пример: `example.com`, что приведет к привязке StarVault как `username@example.com`.

3.3. Разрешение членства в группе

После аутентификации пользователя метод LDAP auth должен знать, как определить, к каким группам принадлежит пользователь. Конфигурация для этого может отличаться в зависимости от вашего сервера LDAP и схемы каталогов. При определении принадлежности к группе используются две основные стратегии: первая - поиск объекта authenticated user и следование атрибуту групп, членом которых он является. Вторая - поиск объектов group, членом которых является аутентифицированный пользователь. Поддерживаются оба метода.

- `groupfilter` (string, необязательно) - шаблон Go, используемый при построении запроса на членство в группе. Шаблон может обращаться к следующим контекстным переменным: [`UserDN`, `Username`]. По умолчанию используется (`|(memberUid={{.Username}})(member={{.UserDN}})(uniqueMember={{.UserDN}})`), который совместим с несколькими распространенными схемами каталогов. Для поддержки разрешения вложенных групп в Active Directory вместо этого используйте следующий запрос: `(&(objectClass=group)(member:1.2.840.113556.1.4.1941:={{.UserDN}}))`.
- `groupdn` (string, обязательно) - база поиска LDAP, используемая для поиска членства в группе. Это может быть корень, содержащий либо группы, либо пользователей.

  Пример: `ou=Groups,dc=example,dc=com`
- `groupdn` (string, обязательно) - база поиска LDAP, используемая для поиска членства в группе. Это может быть корень, содержащий либо группы, либо пользователей.

  Пример: `ou=Groups,dc=example,dc=com`

При использовании аутентифицированного поиска для параметров привязки (см. выше) для поиска группы используется отличительное имя, определенное для `binddn`. В противном случае для поиска группы используется аутентифицированный пользователь.

Для получения большей информации используйте `starvault path-help`.

3.4. Другие

- `username_as_alias` (bool, необязательно) - если установлено значение `true`, это заставляет метод auth использовать имя пользователя, переданное пользователем, в качестве имени псевдонима.
- `max_page_size` (int, необязательно) - если установлено значение больше 0, бэкэнд LDAP будет использовать управление постраничным поиском сервера LDAP для запроса страниц до указанного размера. Это можно использовать, чтобы избежать столкновения

с ограничением максимального размера результатов сервера LDAP. В противном случае бэкэнд LDAP не будет использовать управление постраничным поиском.

4. Примеры

4.1. Сценарий 1

- Сервер LDAP, работает на `ldap.example.com`, порт 389.
- Сервер поддерживает команду `STARTTLS` для запуска шифрования на стандартном порту.
- Сертификат CA хранится в файле с именем `ldap_ca_cert.pem`
- Сервер Active Directory поддерживает атрибут userPrincipalName. Пользователи идентифицируются как `username@example.com`.
- Группы являются вложенными, мы будем использовать `LDAP_MATCHING_RULE_IN_CHAIN` для просмотра списка.
- Поиск группы начнется в разделе `ou=Groups,dc=example,dc=com`. Для всех объектов группы, расположенных по этому пути, атрибут `member` будет проверен на соответствие аутентифицированному пользователю.
- Имена групп идентифицируются с помощью их атрибута `cn`.

BASH

```
$ starvault write auth/ldap/config \
    url="ldap://ldap.example.com" \
    userdn="ou=Users,dc=example,dc=com" \
    groupdn="ou=Groups,dc=example,dc=com" \
    groupfilter="(&(objectClass=group)(member:1.2.840.113556.1.4.1941:=
{{.UserDN}}))" \
    groupattr="cn" \
    upndomain="example.com" \
    certificate=@ldap_ca_cert.pem \
    insecure_tls=false \
    starttls=true
...
```

4.2. Сценарий 2

- Сервер LDAP, работает на `ldap.example.com`, порт 389.
- Сервер поддерживает команду `STARTTLS` для запуска шифрования на стандартном порту.
- Сертификат CA хранится в файле с именем `ldap_ca_cert.pem`

- Сервер не разрешает анонимные привязки для выполнения поиска пользователей.
- Для поиска используется учетная запись привязки `cn=starvault,ou=users,dc=example,dc=com` с паролем `My$ecrt3tP4ss`.
- Объекты пользователя находятся в разделе `ou=Users,dc=example,dc=com`.
- Имя пользователя, передаваемое в хранилище при аутентификации, сопоставляется с атрибутом `sAMAccountName`.
- Принадлежность к группе будет определяться с помощью атрибута `memberOf` объектов user. Поиск начнется в разделе `ou=Users,dc=example,dc=com`.

```bash
$ starvault write auth/ldap/config \
    url="ldap://ldap.example.com" \
    userattr=sAMAccountName \
    userdn="ou=Users,dc=example,dc=com" \
    groupdn="ou=Users,dc=example,dc=com" \
    groupfilter="(&(objectClass=person)(uid={{.Username}}))" \
    groupattr="memberOf" \
    binddn="cn=starvault,ou=users,dc=example,dc=com" \
    bindpass='My$ecrt3tP4ss' \
    certificate=@ldap_ca_cert.pem \
    insecure_tls=false \
    starttls=true
...
```

4.3. Сценарий 3

- Сервер LDAP работает на `ldap.example.com`, порт 636 (LDAPS)
- Сертификат CA хранится в файле с именем `ldap_ca_cert.pem`
- Пользовательские объекты находятся в разделе `ou=Users,dc=example,dc=com`.
- Имя пользователя, передаваемое в хранилище при аутентификации, сопоставляется с атрибутом `uid`.
- Имя пользователя для привязки будет автоматически определено с помощью анонимной привязки.
- Принадлежность к группе будет определяться с помощью любого из атрибутов `memberUid`, `member member` или `uniqueMember`. Этот поиск начнется в разделе `ou=Groups,dc=example,dc=com`.
- Названия групп определяются с помощью атрибута `cn`.

```bash
$ starvault write auth/ldap/config \
    url="ldaps://ldap.example.com" \
    userattr="uid" \
    userdn="ou=Users,dc=example,dc=com" \
```

```
    discoverdn=true \
    groupdn="ou=Groups,dc=example,dc=com" \
    certificate=@ldap_ca_cert.pem \
    insecure_tls=false \
    starttls=true
...
```

5. Группа LDAP → разработка политики

Далее мы хотим создать сопоставление группы LDAP с политикой StarVault:

BASH

```
$ starvault write auth/ldap/groups/scientists policies=foo,bar
```

Это сопоставляет LDAP-группу "scientists" с политиками StarVault "foo" и "bar". Мы также можем добавить определенных пользователей LDAP в дополнительные (потенциально не-LDAP) группы. Обратите внимание, что политики могут быть указаны и для пользователей LDAP.

BASH

```
$ starvault write auth/ldap/groups/engineers policies=foobar
$ starvault write auth/ldap/users/tesla groups=engineers policies=zoobar
```

Это добавляет пользователя LDAP "tesla" в группу "engineers", которая сопоставлена с политикой "foobar" StarVault. Сам пользователь "tesla" связан с политикой "zoobar".

Наконец, мы можем проверить это, пройдя аутентификацию:

BASH

```
$ starvault login -method=ldap username=tesla
Password (will be hidden):
Successfully authenticated! The policies that are associated
with this token are listed below:

default, foobar, zoobar
```

6. Примечание по сопоставлению политик

Следует отметить, что сопоставление пользователь → политика происходит во время создания токена. И изменения в членстве группы на LDAP-сервере не повлияют на токены, которые уже были предоставлены. Чтобы увидеть эти изменения, необходимо отозвать старые токены и попросить пользователя пройти повторную аутентификацию.

7. Блокировка пользователя

Если пользователь несколько раз подряд предоставит неверные учетные данные, StarVault на некоторое время прекратит попытки проверить его учетные данные, а вместо этого сразу же выдаст ошибку об отказе в разрешении. Мы называем такое поведение "блокировкой пользователя". Время, на которое пользователь будет заблокирован, называется "длительностью блокировки". Пользователь сможет войти в систему после истечения срока блокировки. Количество неудачных попыток входа, после которых пользователь будет заблокирован, называется "порог блокировки". Счетчик порога блокировки обнуляется через несколько минут без попыток входа или при успешной попытке входа. Время, в течение которого счетчик будет обнулен после отсутствия попыток входа, называется "сброс счетчика блокировки". Это позволяет предотвратить как автоматические, так и целевые запросы, то есть атаки на угадывание 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:

- его можно отключить глобально с помощью переменной окружения `VAULT_DISABLE_USER_LOCKOUT`.
- его можно отключить для всех поддерживаемых методов аутентификации (ldap, user-pass и approle) или для определенного поддерживаемого метода аутентификации с помощью параметра `disable_lockout` в строке `user_lockout` в конфигурационном файле. Более подробная информация приведена в разделе Конфигурация блокировки пользователей.
- его можно отключить для конкретного монтирования с помощью команды "auth tune". Более подробную информацию можно найти в команде auth tune или auth tune api.

Эта функция поддерживается только методами userpass, ldap и approle auth.

8. API

Метод LDAP auth имеет полноценный HTTP API. Более подробная информация приведена в разделе API метода LDAP auth.

Методы аутентификации. Login MFA

StarVault поддерживает многофакторную аутентификацию (MFA) для проверки подлинности метода аутентификации с использованием различных типов аутентификации. Login MFA построена поверх системы идентификации StarVault.

1. Типы MFA

MFA в StarVault включает следующие типы входа:

> Метод аутентификации Token не может быть настроен с помощью встроенной в StarVault функции Login MFA.

- `Одноразовый пароль на основе времени (TOTP)` — если настроен и включен на пути входа в систему, то при запросе на вход в API потребуется ввести код TOTP вместе с токеном StarVault. Пароль будет проверен на соответствие ключу TOTP, присутствующему в идентификаторе вызывающего абонента в StarVault.
- `Duo` — если Duo push настроен и включен на пути входа, то зарегистрированное устройство пользователя получит push-уведомление, чтобы одобрить или запретить доступ к API. Имя пользователя Duo будет получено из псевдонима вызывающего пользователя. Обратите внимание, что Duo также можно настроить на использование кодов для аутентификации.
- `PingID` — если PingID push настроен и включен на пути входа, зарегистрированное устройство пользователя получит push-уведомление, чтобы одобрить или отклонить доступ к API. Имя пользователя PingID будет получено из псевдонима вызывающего абонента.

2. Процедура Login MFA

> Встроенная в StarVault функция Login MFA по умолчанию не защищает от грубого взлома паролей TOTP. Рекомендуем применять ограничения скорости для каждого клиента к соответствующим путям входа и/или mfa (например, `/sys/mfa/validate`). Внешние методы MFA (`Duo`, `Ping`) могут уже предоставлять настраиваемое ограничение скорости. Ограничение скорости для путей Login MFA применяется по умолчанию.

MFA для входа в систему может быть настроена для защиты дальнейшей аутентификации по методу аутентификации. Чтобы включить MFA для входа в систему, необходимо настроить метод MFA. Подробную информацию о настройке метода MFA см. в разделе API

Login MFA. После того как метод MFA настроен, оператор может настроить применение MFA, используя полученный уникальный идентификатор метода MFA. Подробнее о том, как настроить конфигурацию внедрения MFA, см. в Login MFA Enforcement API. MFA может быть применена к сущности, группе сущностей, определенному аксессору метода аутентификации или типу метода аутентификации. Запрос на вход, соответствующий любым ограничениям MFA, перед аутентификацией проходит дополнительную проверку MFA, например, одноразовый код доступа.

Существует два способа проверки запроса на вход, который подлежит проверке MFA.

2.1. Вход за одну итерацию (Single-Phase login)

При входе в систему за одну итерацию необходимая информация MFA включается в запрос на вход с помощью заголовка `X-Vault-MFA`. В этом случае проверка MFA выполняется как часть запроса на вход.

Учетные данные MFA извлекаются из HTTP-заголовка `X-Vault-MFA`. Поддерживаются два формата заголовка: `mfa_method_id[:passcode]` или `mfa_method_id[:passcode=<passcode>]`. Элемент в `[]` является необязательным. Если необходимо подтвердить несколько методов MFA, пользователь может передать несколько HTTP-заголовков `X-Vault-MFA`.

2.1.1. Образец запроса

BASH

```
curl \
    --header "X-Vault-Token: ..." \
    --header "X-Vault-MFA: d16fd3c2-50de-0b9b-eed3-0301dadeca10:695452" \
    http://127.0.0.1:8200/v1/auth/userpass/login/alice
```

Если метод MFA не требует ввода пароля, то заголовок MFA запроса на вход в систему содержит только идентификатор метода.

BASH

```
 $ curl \
     --header "X-Vault-Token: ..." \
     --header "X-Vault-MFA: d16fd3c2-50de-0b9b-eed3-0301dadeca10" \
     http://127.0.0.1:8200/v1/auth/userpass/login/alice
```

В StarVault оператор может задать имя для метода MFA. Это имя должно быть уникальным. Имя метода MFA может быть использовано в заголовке MFA.

BASH

```
curl \
    --header "X-Vault-Token: ..." \
    --header "X-Vault-MFA: sample_mfa_method_name:695452" \
    http://127.0.0.1:8200/v1/auth/userpass/login/alice
```

2.2. Вход за две итерации (Two-Phase login)

Более традиционным и распространенным методом MFA является механизм двух запросов, также называемый Two-phase Login MFA. При входе в систему за две итерации в заголовок `X-Vault-MFA` в запросе не указывается. В этом случае после отправки обычного запроса на вход пользователь получает ответ auth, в котором содержатся требования MFA. Требования MFA содержат идентификатор запроса MFA, который идентифицирует запрос на вход в систему, нуждающийся в проверке. Кроме того, требования MFA содержат ограничения MFA, которые определяют необходимые типы MFA для проверки запроса, соответствующие идентификаторы методов и булево значение, показывающее, использует ли метод MFA коды или нет. Ограничения MFA образуют вложенную карту в MFA Requirement и представляют все принудительные меры MFA, которые соответствуют запросу на вход. Хотя приведенный ниже пример относится к логину userpass, обратите внимание, что это может повлиять на ответ на вход в систему при любом аутентификационном монтировании, защищенном проверкой MFA.

2.2.1. Образец двухфазного ответа на вход в систему


```
{
  "request_id": "1044c151-13ea-1cf5-f6ed-000c42efd477",
  "lease_id": "",
  "lease_duration": 0,
  "renewable": false,
  "data": null,
  "warnings": [
    "A login request was issued that is subject to MFA validation. Please make
sure to validate the login by sending another request to mfa/validate endpoint."
  ],
  "auth": {
    "client_token": "",
    "accessor": "",
    "policies": null,
    "token_policies": null,
    "identity_policies": null,
    "metadata": null,
    "orphan": false,
    "entity_id": "",
    "lease_duration": 0,
    "renewable": false,
    "mfa_requirement": {
      "mfa_request_id": "d0c9eec7-6921-8cc0-be62-202b289ef163",
      "mfa_constraints": {
        "enforcementConfigUserpass": {
          "any": [
            {
              "type": "totp",
              "id": "820997b3-110e-c251-7e8b-ff4aa428a6e1",
              "uses_passcode": true,
```

```
            "name": "sample_mfa_method_name",
          }
        ]
      }
    }
  }
}
```

Обратите внимание, что значение `uses_passcode` всегда будет показывать `true` для `TOTP`, и `false` `PingID`. Для метода Duo значение может быть настроено в рамках конфигурации метода с помощью параметра use_passcode. Подробную информацию о том, как настроить булево значение для `Duo`, см. в Duo API.

Чтобы подтвердить запрос на вход с ограничением MFA, пользователь отправляет второй запрос на конечную точку `validate`, содержащий идентификатор запроса MFA и полезную нагрузку MFA. Полезная нагрузка MFA содержит карту идентификаторов методов и связанных с ними учетных данных. Если настроенные методы MFA (`PingID` и `Duo`), не требуют ввода пароля, связанные с ними учетные данные будут представлять собой список с одной пустой строкой.

2.2.2. Образец полезной нагрузки

```
{
  "mfa_request_id": "5879c74a-1418-1948-7be9-97b209d693a7",
  "mfa_payload": {
    "d16fd3c2-50de-0b9b-eed3-0301dadeca10": ["910201"]
  }
}
```

2.2.3. Образец запроса

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

2.2.4. Образец запроса через CLI

Пользователь также может использовать команду CLI write для подтверждения запроса на вход.

BASH

```
starvault write sys/mfa/validate -format=json @payload.json
```

2.2.5. Интерактивный CLI для входа в систему MFA

StarVault поддерживает интерактивный способ аутентификации в методе аутентификации с помощью CLI только в том случае, если запрос на вход подвергается проверке одним методом MFA. В этой ситуации, если метод MFA настроен на использование кодов, после отправки обычного запроса на вход пользователю предлагается ввести код. При успешной проверке MFA возвращается клиентский токен. Если настроенные методы MFA — PingID и Duo — не требуют ввода пароля и имеют внеполосные механизмы проверки дополнительного фактора, пользователю сообщается о необходимости проверить приложение аутентификатора. Это избавляет пользователя от необходимости отправлять второй запрос отдельно для проверки запроса на вход. Чтобы отключить интерактивный вход, пользователю необходимо передать флаг `non-interactive` в запросе на вход.

```bash
starvault write -non-interactive sys/mfa/validate -format=json @payload.json
```

Чтобы начать работу с Login MFA, обратитесь к учебному пособию по Login MFA.

2.3. Ограничение скорости проверки пароля TOTP

В StarVault по умолчанию включено ограничение скорости входа в MFA. По умолчанию StarVault допускает 5 последовательных неудачных попыток проверки пароля TOTP. Это значение также можно настроить, добавив `max_validation_attempts` в конфигурацию TOTP. Если количество неудачных попыток проверки пароля TOTP превысит заданное значение, пользователю придется подождать, пока не будет получен новый код TOTP.

3. FAQ по Login MFA

В этом разделе FAQ собраны часто задаваемые вопросы о функции Login MFA.

- Вопрос: Какие различные рабочие процессы MFA доступны как пользователю StarVault, и чем они отличаются?
- Вопрос: Что такое однофазный MFA по сравнению с двухфазным MFA?
- Вопрос: Какие существуют конечные точки MFA API в StarVault MFA для входа в систему?
- Вопрос: Какие существуют способы настройки различных рабочих процессов MFA?
- Вопрос: Какой механизм MFA используется с различными рабочими процессами MFA в StarVault?
- Вопрос: Если используется Vault Agent, возникнут ли у проблемы с MFA?

3.1. Вопрос: Какие различные рабочие процессы MFA доступны как пользователю StarVault, и чем они

отличаются?

Поток работ MFA	Что он делает?	Кто управляет MFA?
Login MFA	MFA в StarVault обеспечивает MFA при входе в систему. Поддерживается вход через CLI, API и пользовательский интерфейс	MFA управляется StarVault

3.2. Вопрос: Что такое однофазный MFA по сравнению с двухфазным MFA?

- Вход за одну итерацию (Single-Phase login) — Это механизм одного запроса, в котором необходимая информация MFA, например идентификатор метода MFA, предоставляется через заголовок `X-Vault-MFA` в одном MFA-запросе, который используется для аутентификации в Vault.

  Если настроенные методы MFA требуют пароля, его необходимо указать в запросе, как, например, в случае `TOTP` или `Duo`. Если настроенные методы MFA, такие как `PingID` или `Duo`, не требуют пароля и имеют внеполосные механизмы проверки дополнительного фактора, Vault отправит запрос в API другого сервиса, чтобы определить, был ли запрос MFA уже проверен.

- Вход за две итерации (Two-Phase login) — метод MFA, который используется более традиционно/
  - Пароль MFA, необходимый для настроенного метода MFA, не предоставляется в заголовке запроса на вход, который ограничен MFA. Вместо этого пользователь сначала проходит аутентификацию в методе аутентификации, и после успешной аутентификации в методе аутентификации пользователю возвращается требование MFA. Требование MFA содержит идентификатор запроса MFA и ограничения, применимые к MFA, как настроено оператором.
  - Затем пользователь должен сделать второй запрос к новой конечной точке `sys/mfa/validate`, указав в запросе MFA RequestID и полезную нагрузку MFA, включающую код доступа MFA methodIDs (если применимо). Если проверка MFA пройдет, новый токен Vault будет сохранен и возвращен пользователю в ответе, как и обычный токен Vault, созданный с помощью метода авторизации без ограничений MFA.

3.3. Вопрос: Какие существуют конечные точки MFA API в StarVault MFA для входа в систему?

Эта функция опирается на следующие конечные точки конфигурации MFA: `identity/mfa/method`, `identity/mfa/login-enforcement` и `sys/mfa/validate`. Более подробную информацию см. в документации.

3.4. Вопрос: Какие существуют способы настройки различных рабочих процессов MFA?

Поток работ MFA	Методы конфигурирования	Детали
Login MFA	CLI/API/UI.	Настраивается с помощью конечных точек `identity/mfa/method`, затем передаются идентификаторы методов в конечную точку `identity/mfa/login-enforcement`. Поддерживаемые методы MFA: `TOTP`, `Okta`, `Duo`, `PingID`.

3.5. Вопрос: Какой механизм MFA используется с различными рабочими процессами MFA в StarVault?

Поток работ MFA	UI	CLI/API	Single-Phase	Two-Phase
Login MFA	Поддерживается	Поддерживается. Можете выбрать вход за одну итерацию MFA, предоставив заголовок X-Vault-MFA. При отсутствии этого заголовка используется вход за две итерации MFA	N/A	Поддерживается

3.6. Вопрос: Если используется StarVault Agent, возникнут ли у проблемы с MFA?

Агент StarVault Agent не должен использовать MFA для аутентификации в StarVault; он должен быть в состоянии успешно передавать запросы с заголовками, связанными с MFA, в StarVault.