Блокировка пользователей

1. Общие сведения о блокировке пользователей

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 доступе.

Время, на которое пользователь будет заблокирован, называется **длительностью блокировки (lockout duration)**. Пользователь сможет войти в систему после истечения срока блокировки. Количество неудачных попыток входа, после которых пользователь будет заблокирован, называется **порог блокировки (lockout threshold)**. 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 (lockout counter reset)**. Это позволяет противостоять как автоматизированным, так и целенаправленным запросам, то есть атакам на угадывание пароля.

Функция блокировки пользователя включена по умолчанию. Значения по умолчанию оставляют:

- **порог блокировки** - 5 попыток
- **длительность блокировки** - 15 минут
- **сброс счетчика блокировки** - 15 минут.

Функцию блокировки пользователя можно отключить следующими способами:

- Глобально с использованием переменной окружения **VAULT_DISABLE_USER_LOCKOUT**.
- Для всех поддерживаемых методов аутентификации (ldap, userpass и approle) или для конкретного поддерживаемого метода аутентификации с использованием параметра `disable_lockout` внутри секции `user_lockout` в конфигурационном файле.
- Для конкретной точки монтирования метода аутентификации с использованием команды **auth tune**.

2. Блок конфигурации user_lockout

Блок конфигурации `user_lockout` определяет различные настройки поведения блокировки пользователя при неудачных попытках входа в StarVault. Они могут быть

настроены для всех поддерживаемых методов аутентификации (userpass, ldap и approle) с использованием общего имени секции `user_lockout "all"` или для конкретного метода аутентификации, используя имя этого метода в секции.

Поддерживаемые имена секций `user_lockout` включают **all**, **userpass**, **ldap** и **approle**.

Конфигурации для конкретного метода аутентификации имеют приоритет над настройками, указанными для всех методов аутентификации с использованием общего имени секции `user_lockout "all"` в конфигурационном файле.

2.1. Параметры секции user_lockout

lockout_threshold

Строковый параметр. Указывает количество неудачных попыток входа в систему, после чего пользователь будет заблокирован.

lockout_duration

Строковый параметр. Указывает длительность блокировки пользователя. Значение указывается с использованием суффикса времени, например "30s" (30 секунд) или "1h" (1 час).

lockout_counter_reset

Строковый параметр. Определяет интервал времени, после истечения которого счетчик блокировки сбрасывается при отсутствии неудачных попыток входа в систему. Значение указывается с использованием суффикса времени, например "30s" (30 секунд) или "1h" (1 час).

disable_lockout

Логический параметр. Отключает функцию блокировки пользователя, если установить в значение `true`.

2.2. Пример конфигурации

```
user_lockout "all" {
  lockout_duration = "10m"
  lockout_counter_reset = "10m"
}

user_lockout "userpass" {
  lockout_threshold = "25"
  lockout_duration = "5m"
}

user_lockout "ldap" {
```

```
 disable_lockout = "true"
}
```

В приведенном выше примере функция блокировки пользователя будет отключена для методов аутентификации **ldap**. Для методов аутентификации **userpass** порог блокировки будет равен **25**, длительность блокировки **5 минут**, сброс счетчика блокировки **10 минут** (используется значение **all**, так как не определено в секции **userpass**). Методы аутентификации **approle** будут иметь порог блокировки **5** (используется значение по умолчанию, так как секция для **approle** отсутствует, а в **all** этот параметр не задан), длительность блокировки **10 минут** и сброс счетчика блокировки **10 минут**.

Блок конфигурации listener

Блок конфигурации **listener** настраивает адреса и порты, на которыхStarVault будет отвечать на запросы. На данный момент существует два типа слушателей:

- TCP
- Unix Domain Socket

1. TCP

TCP слушатель настраивает StarVault на прослушивание TCP-адреса/порта.

```
listener "tcp" {
  address = "127.0.0.1:8200"
}
```

Блок конфигурации `listener` может быть указана более одного раза, чтобы StarVault прослушивал несколько интерфейсов. Если вы настраиваете несколько слушателей, вам также необходимо указать `api_addr` и `cluster_addr`, чтобы StarVault анонсировал другим узлам корректный адрес.

1.1. Сокрытие конфиденциальных данных для неаутентифицированных конечных точек

Неаутентифицированные конечные точки API могут возвращать следующую конфиденциальную информацию:

- Номер версии StarVault
- Дату сборки бинарного файла StarVault
- Имя кластера StarVault
- IP-адрес узлов в кластере

StarVault предлагает возможность настроить каждую секцию `tcp listener` таким образом, чтобы, при необходимости, эти значения были удалены из ответов.

Следующие конечные точки API поддерживают удаление информации на основе конфигурации секции listener:

- **/sys/health**

- **/sys/leader**
- **/sys/seal-status**

StarVault заменяет удаленную информацию пустой строкой (""). Некоторые API StarVault также опускают ключи в ответе, когда соответствующее значение пустое ("").

> Удаление значений влияет на ответы для всех клиентов API.
>
> Клиентская командная строка StarVault и пользовательский интерфейс используют ответы API StarVault. В результате настройки удаления будут применяться к выводу в CLI и пользовательском интерфейсе, а также к прямым вызовам API.

1.2. Пользовательские заголовки ответов

StarVault поддерживает определение пользовательских HTTP-заголовков ответов для корневого пути (`/`) и также для конечных точек API (`/v1/` **). Заголовки определяются на основе возвращаемого статусного кода. Например, пользователь может определить список пользовательских заголовков ответов для статусного кода *200** и другой список пользовательских заголовков ответов для статусного кода **307**. Существует конечная точка API **/sys/config/ui**, которая позволяет пользователям устанавливать специфические для UI пользовательские заголовки. Если заголовок настроен в конфигурационном файле, его нельзя перенастроить через конечную точку API **/sys/config/ui**. В случаях, когда необходимо изменить значение пользовательского заголовка или удалить пользовательский заголовок, должен быть соответственно изменен файл конфигурации StarVault, а в процесс StarVault необходимо отправить сигнал SIGHUP.

Если заголовок определен в конфигурационном файле и тот же заголовок используется внутренними процессами StarVault, настроенный заголовок не принимается. Например, пользовательский заголовок с префиксом `X-Vault-` не будет принят. При запуске в журналах StarVault будет зарегистрировано сообщение, указывающее, что заголовок с префиксом `X-Vault-` не принят.

1.2.1. Порядок приоритетов

Если один и тот же заголовок настроен как в конфигурационном файле, так и в конечной точке API **/sys/config/ui**, приоритет имеет заголовок в конфигурационном файле. Например, заголовок `Content-Security-Policy` по умолчанию определен в конечной точке API **/sys/config/ui**. Если этот заголовок также определен в конфигурационном файле, то значение из конфигурационного файла устанавливается в заголовках ответа вместо значения по умолчанию в конечной точке **/sys/config/ui**.

1.3. Параметры слушателя TCP

address

Строковый параметр в формате `ip-адрес:порт`. Указывает адрес и порт, назначенные для прослушивания. Он может быть динамически определен с помощью шаблона go-sockaddr, который разрешается во время выполнения.

cluster_address

Строковый параметр в формате `ip-адрес:порт`. Указывает адрес и порт, назначенные для прослушивания запросов типа сервер-сервер в кластере. По умолчанию это значение на один порт выше, чем значение `address`. Обычно это не требуется устанавливать, но может быть полезным в случаях, когда серверы StarVault изолированы друг от друга таким образом, что им нужно перескакивать через TCP-балансировщик нагрузки или использовать какую-то другую схему для связи. Это может быть динамически определено с помощью шаблона go-sockaddr, который разрешается во время выполнения.

chroot_namespace

Строковый параметр. Указывает альтернативное пространство имен верхнего уровня для слушателя. StarVault добавляет пространства имен, указанные в заголовке `X-Vault-Namespace` или поле `-namespace` в команде CLI, к пространству имен верхнего уровня, чтобы определить полный путь пространства имен для запроса. Например, если `chroot_namespace` установлено в `admin` и заголовок `X-Vault-Namespace` это `ns1`, полный путь пространства имен будет admin/ns1. Вызовы к слушателю завершатся ошибкой **4XX**, если пространство имен верхнего уровня, указанное для `chroot_namespace`, не существует.

http_idle_timeout

Строковый параметр. Указывает максимальное время ожидания следующего запроса при включении режима **keep-alives**. Если `http_idle_timeout` равно нулю, используется значение `http_read_timeout`. Если оба значения равны нулю, используется значение `http_read_header_timeout`. Оно указывается с помощью суффикса времени, например "30s" или "1h".

http_read_header_timeout

Строковый параметр. Указывает количество времени, отведенное на чтение заголовков запроса. Оно задается с помощью суффикса времени, например "30s" или "1h".

http_read_timeout

Строковый параметр. Указывает максимальную продолжительность чтения всего запроса, включая тело. Она задается с помощью суффикса времени, например "30s" или "1h".

http_write_timeout

Строковый параметр. Указывает максимальную продолжительность записи ответа и сбрасывается каждый раз, когда считывается заголовок нового запроса. Значение по умолчанию **0** означает неограниченное время. Это значение задается с помощью суффикса времени, например "30s" или "1h".

max_request_size

Целоцисленный параметр. Указывает максимально допустимый размер запроса, в байтах. По умолчанию равен 32 МБ, если не установлен или установлен в **0**. Указание числа меньше **0** полностью отключает ограничение.

max_request_duration

Строковый параметр. Указывает максимальную продолжительность обработки запроса, после которой StarVault отменяет запрос. Этот параметр отменяет значение `default_max_request_duration` для данного слушателя.

proxy_protocol_behavior

Строковый параметр. Включает поведение протокола PROXY версии 1 для слушателя. Принимаемые значения:

- `use_always` - всегда будет использоваться IP-адрес клиента.
- `allow_authorized` - если исходный IP-адрес находится в списке `proxy_protocol_authorized_addrs`, будет использоваться IP-адрес клиента. Если исходный IP не в списке, будет использоваться исходный IP-адрес.
- `deny_unauthorized` - трафик будет отклонен, если исходный IP-адрес не находится в списке `proxy_protocol_authorized_addrs`.

proxy_protocol_authorized_addrs

Определяет список разрешенных IP-адресов источников для использования с протоколом PROXY.

- Не требуется, если для параметра `proxy_protocol_behavior` установлено значение `use_always`.
- Список IP-адресов источников может быть указан в формате строки или списка. При указании в формате строки значения должны быть разделены запятыми.
- Должен быть указан хотя бы один IP-адрес источника, `proxy_protocol_authorized_addrs` не может быть пустым списком или строкой.

redact_addresses

Логический параметр. Когда установлено в значение `true`, скрывает значения `leader_address` и `cluster_leader_address` в соответствующих ответах API.

redact_cluster_name

Логический параметр. Когда установлено в значение `true`, скрывает значения `cluster_name` в соответствующих ответах API.

redact_version

Логический параметр. Когда установлено в значение `true`, скрывает значения `version` и `build_date` в соответствующих ответах API.

tls_disable

Логический параметр. Указывает, будет ли отключен TLS. StarVault использует TLS по умолчанию, поэтому необходимо явно отключить TLS, чтобы отказаться от небезопасной связи. Отключение TLS может привести к отключению некоторых функций пользовательского интерфейса.

tls_cert_file

Строковый параметр. Указывает путь к сертификату для TLS. Требуется файл в кодировке PEM. Чтобы настроить слушателя на использование сертификата ЦС, нужно объединить основной сертификат и сертификат ЦС вместе. Основной сертификат должен находиться в начале объединенного файла. Путь, указанный в данном параметре, используется при запуске StarVault; изменение этого значения во время работы StarVault не окажет никакого влияния.

tls_key_file

Строковый параметр. Указывает путь к закрытому ключу для сертификата. Требуется файл в кодировке PEM. Если файл ключа зашифрован, при запуске сервера будет запрошен ввод парольной фразы. Парольная фраза должна оставаться неизменной между файлами ключей при перезагрузке конфигурации с помощью SIGHUP. Путь, указанный в данном параметре, используется при запуске StarVault; изменение этого значения во время работы StarVault не окажет никакого влияния.

tls_min_version

Строковый параметр. Указывает минимальную поддерживаемую версию TLS. Принимаются следующие значения: "tls10", "tls11", "tls12" или "tls13".

TLS 1.1 и ниже (значения tls10 и tls11 для параметров `tls_min_version` и `tls_max_version`) считаются небезопасными.

tls_max_version

Строковый параметр. Указывает максимальную поддерживаемую версию TLS. Принимаются следующие значения: "tls10", "tls11", "tls12" или "tls13".

TLS 1.1 и ниже (значения tls10 и tls11 для параметров `tls_min_version` и `tls_max_version`) считаются небезопасными.

tls_cipher_suites

Строковый параметр. Указывает список поддерживаемых наборов шифров в виде списка значений, разделенных запятыми. Список всех доступных наборов шифров доступен в документации по TLS для Golang.

Go консультируется со списком `tls_cipher_suites` только для TLSv1.2 и более ранних версий; порядок шифров не важен. Для того чтобы этот параметр был эффективен, свойство `tls_max_version` должно быть установлено в `tls12`, чтобы предотвратить согласование TLSv1.3, что не рекомендуется. Для получения дополнительной информации об этом и других изменениях, связанных с TLS, смотрите пост о TLS от Go.

tls_require_and_verify_client_cert

Логический параметр. Включает аутентификацию клиента для этого слушателя; слушателю потребуется представленный клиентский сертификат, успешно проверенный на системных ЦС.

tls_client_ca_file

Строковый параметр. Файл сертификата ЦС в кодировке PEM, используемый для проверки подлинности клиента.

tls_disable_client_certs

Логический параметр. Отключает аутентификацию клиента для этого слушателя. Поведение по умолчанию (значение установлено в `false`) предполагает, что StarVault запрашивает сертификаты аутентификации клиента, когда они доступны.

Поля `tls_disable_client_certs` и `tls_require_and_verify_client_cert` в секции `listener` конфигурации сервера StarVault являются взаимоисключающими. Пожалуйста, убедитесь, что они оба не установлены в значение `true`. Проверка клиента TLS остаётся необязательной с настройками по умолчанию и не является обязательной.

x_forwarded_for_authorized_addrs

Строковый параметр. Может быть указан в формате строки со списком значений, разделенных запятыми или в формате JSON-массива. Указывает список CIDR исходных IP-адресов, для которых заголовок `X-Forwarded-For` будет доверенным. Это включает поддержку `X-Forwarded-For`. Например, если StarVault получает соединения с IP-адреса балансировщика нагрузки **1.2.3.4**, добавление **1.2.3.4** в `x_forwarded_for_authorized_addrs` приведет к тому, что поле `remote_address` в журнале аудита будет заполнено IP-адресом подключающегося клиента, например **3.4.5.6**. Обратите внимание, что это требует, чтобы балансировщик нагрузки отправлял IP-адрес подключающегося клиента в заголовке `X-Forwarded-For`.

x_forwarded_for_hop_skips

Строковый параметр. Количество адресов, которые будут пропущены с конца набора переходов. Например, для значения заголовка `1.2.3.4, 2.3.4.5, 3.4.5.6, 4.5.6.7`, если это значение установлено в "1", адрес, который будет использоваться как IP-адрес исходящего клиента, будет **3.4.5.6**.

x_forwarded_for_reject_not_authorized

Логический параметр. Если установлено значение `false`, то при наличии заголовка `X-Forwarded-For` в соединении с неавторизованного адреса заголовок будет проигнорирован, а клиентское соединение будет использоваться как есть, вместо того чтобы отклонить клиентское соединение.

x_forwarded_for_reject_not_present

Логический параметр. Если установлено значение `false`, то если заголовок `X-Forwarded-For` отсутствует или пуст, адрес клиента будет использоваться как есть, вместо того чтобы соединение с клиентом было отклонено.

disable_replication_status_endpoints

Логический параметр. При установке значения `true` отключает конечные точки статуса репликации для настроенного слушателя.

1.3.1. Параметры telemetry

unauthenticated_metrics_access

Логический параметр. При установке значения `true` разрешается неаутентифицированный доступ к конечной точке **/v1/sys/metrics**.

1.3.2. Параметры profiling

unauthenticated_pprof_access

Логический параметр. При установке значения `true` разрешается неаутентифицированный доступ к конечной точке **/v1/sys/pprof**.

1.3.3. Параметры inflight_requests_logging

unauthenticated_in_flight_requests_access

Логический параметр. При установке значения `true` разрешается неаутентифицированный доступ к конечной точке **/v1/sys/in-flight-req**.

1.3.4. Параметры custom_response_headers

default

Список сопоставлений типа:

```
{
  "ключ1" = ["значение1", "значение 2", ...],
  "ключ2" = ["значение1", "значение 2", ...],
}
```

Позволяет сопоставить имена заголовков по умолчанию с массивом значений. Заголовки по умолчанию устанавливаются на всех конечных точках независимо от значения статусного кода.

Подробнее см. в примерах.

<specific status code>

Список сопоставлений типа:

```
{
  "ключ1" = ["значение1", "значение 2", ...],
  "ключ2" = ["значение1", "значение 2", ...],
}
```

Позволяет сопоставить имена заголовков с массивом значений. Заголовки, указанные в этой секции устанавливаются для указанных статусных кодов.

Подробнее см. в примерах.

<collective status code>

Список сопоставлений типа:

```
{
  "ключ1" = ["значение1", "значение 2", ...],
  "ключ2" = ["значение1", "значение 2", ...],
}
```

Позволяет сопоставить имена заголовков с массивом значений. Заголовки, указанные в этой секции устанавливаются для статусных кодов, попадающих в указанные группы кодов.

Подробнее см. в примерах.

1.4. Примеры конфигурации секции listener tcp

Пример 1. Указание параметров, необходимых для TLS

```
listener "tcp" {
  tls_cert_file = "/etc/certs/tls.crt"
  tls_key_file  = "/etc/certs/tls.key"
}
```

В данном примере демонстрируется указание сертификата и ключа для TLS.

Пример 2. Прослушивание на нескольких интерфейсах

```
listener "tcp" {
  address = "127.0.0.1:8200"
}
```

```
listener "tcp" {
  address = "10.0.0.5:8200"
}

# Advertise the non-loopback interface
api_addr = "https://10.0.0.5:8200"
cluster_addr = "https://10.0.0.5:8201"
```

В этом примере показано прослушивание StarVault на частном интерфейсе, а также localhost.

Пример 3. Разрешение неаутентифицированного доступа к метрикам

```
listener "tcp" {
  telemetry {
    unauthenticated_metrics_access = true
  }
}
```

Пример 4. Разрешение неаутентифицированного доступа к профилированию

```
listener "tcp" {
  profiling {
    unauthenticated_pprof_access = true
    unauthenticated_in_flight_request_access = true
  }
}
```

Пример 5. Настройка пользовательских заголовков ответов

```
listener "tcp" {
  custom_response_headers {
    "default" = {
      "Strict-Transport-Security" = ["max-age=31536000","includeSubDomains"],
      "Content-Security-Policy" = ["connect-src
https://clusterA.vault.external/"],
      "X-Custom-Header" = ["Custom Header Default Value"],
    },
    "2xx" = {
      "Content-Security-Policy" = ["connect-src
https://clusterB.vault.external/"],
      "X-Custom-Header" = ["Custom Header Value 1", "Custom Header Value 2"],
    },
    "301" = {
      "Strict-Transport-Security" = ["max-age=31536000"],
```

```
      "Content-Security-Policy" = ["connect-src
https://clusterC.vault.external/"],
    },
  }
}
```

Этот пример показывает настройку пользовательских HTTP-заголовков ответа. Операторы могут настроить подсекцию `custom_response_headers` в секции `listener` для установки пользовательских HTTP-заголовков, соответствующих их приложениям. Примеры таких заголовков - `Strict-Transport-Security` и `Content-Security-Policy`, которые являются известными HTTP-заголовками и могут быть настроены для усиления безопасности приложения, взаимодействующего с конечными точками StarVault. Обратите внимание, что сканеры уязвимостей часто исследуют такие связанные с безопасностью HTTP-заголовки. Кроме того, могут быть настроены и специфические для приложения пользовательские заголовки. Например, в приведенном выше примере настроен `X-Custom-Header`.

В ситуациях, когда заголовок определен в нескольких подразделах статусных кодов, будет возвращен заголовок, соответствующий наиболее конкретному коду ответа. Например, с примером конфигурации ниже, ответ 307 вернет значение заголовка Custom для 307, в то время как ответ 306 вернет значение заголовка Custom для 3xx.

```
listener "tcp" {
  custom_response_headers {
    "default" = {
       "X-Custom-Header" = ["default Custom header value"]
    },
    "3xx" = {
       "X-Custom-Header" = ["3xx Custom header value"]
    },
    "307" = {
       "X-Custom-Header" = ["307 Custom header value"]
    }
  }
}
```

Пример 6. Прослушивание на всех IPv4 и IPv6 интерфейсах

В этом примере StarVault прослушивает все интерфейсы IPv4 и IPv6, включая localhost.

```
listener "tcp" {
  address         = "[::]:8200"
  cluster_address = "[::]:8201"
}
```

Пример 7. Прослушивание на определённых IPv6 адресах

Этот пример демонстрирует настройку использования только IPv6 с привязкой к интерфейсу с IP-адресом: `2001:1c04:90d:1c00:a00:27ff:fefa:58ec`.

```
listener "tcp" {
  address         = "[2001:1c04:90d:1c00:a00:27ff:fefa:58ec]:8200"
  cluster_address = "[2001:1c04:90d:1c00:a00:27ff:fefa:58ec]:8201"
}

# Объявление не loopback интерфейса
api_addr = "https://[2001:1c04:90d:1c00:a00:27ff:fefa:58ec]:8200"
cluster_addr = "https://[2001:1c04:90d:1c00:a00:27ff:fefa:58ec]:8201"
```

1.5. Примеры скрытия информации

В примере ниже представлена конфигурация с использованием параметров `redact_addresses`, `redact_cluster_name` и `redact_version` для скрытия информации в ответах.

```
ui            = true
cluster_addr  = "https://127.0.0.1:8201"
api_addr      = "https://127.0.0.1:8200"
disable_mlock = true

storage "raft" {
  path = "/path/to/raft/data"
  node_id = "raft_node_1"
}

listener "tcp" {
  address             = "127.0.0.1:8200",
  tls_cert_file = "/path/to/full-chain.pem"
  tls_key_file  = "/path/to/private-key.pem"
  redact_addresses    = "true"
  redact_cluster_name = "true"
  redact_version      = "true"
}

telemetry {
  statsite_address = "127.0.0.1:8125"
  disable_hostname = true
}
```

1.5.1. Результаты применения параметров скрытия

API: /sys/health

В следующем вызове **/sys/health/** `cluster_name` и `version` скрыты. Поле `cluster_name` полностью исключено из ответа, а `version` представлено пустой строкой ("").

```
$ curl -s https://127.0.0.1:8200/v1/sys/health | jq

{
  "initialized": true,
  "sealed": false,
  "standby": false,
  "performance_standby": false,
  "replication_performance_mode": "disabled",
  "replication_dr_mode": "disabled",
  "server_time_utc": 1715935559,
  "version": "",
  "cluster_id": "be574716-e7e9-a950-ee34-d62d56cd6d4a"
}
```

API: sys/leader

В следующем вызове **/sys/leader/** `leader_address` и `leader_cluster_address` скрыты и установлены в пустую строку ("").

```
curl -s https://127.0.0.1:8200/v1/sys/leader | jq

{
  "ha_enabled": true,
  "is_self": true,
  "active_time": "2024-05-13T07:54:20.471072843Z",
  "leader_address": "",
  "leader_cluster_address": "",
  "performance_standby": false,
  "performance_standby_last_remote_wal": 0,
  "raft_committed_index": 78,
  "raft_applied_index": 78
}
```

API: sys/seal-status

В следующем вызове **/sys/seal-status/** `cluster_name`, `build_date` и `version` скрыты. Поле `cluster_name` полностью исключено из ответа, а `build_date` и `version` представлены пустыми строками ("").

```
curl -s https://127.0.0.1:8200/v1/sys/seal-status | jq

{
  "type": "shamir",
  "initialized": true,
  "sealed": false,
```

```
  "t": 3,
  "n": 6,
  "progress": 0,
  "nonce": "",
  "version": "",
  "build_date": "",
  "migration": false,
  "cluster_id": "be574716-e7e9-a950-ee34-d62d56cd6d4a",
  "recovery_seal": false,
  "storage_type": "raft"
}
```

CLI: starvault status

Команда CLI `starvault status` использует конечные точки, которые поддерживают скрытие данных, поэтому в выводе скрываются `Version`, `Build Date`, `HA Cluster` и `Active Node Address`. `Version`, `Build Date`, `HA Cluster` показывают `n/a`, потому что соответствующая конечная точка вернула пустую строку, а `Active Node Address` показывается как `<none>`, потому что он был опущен в ответе API.

```
starvault status

Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  false
Total Shares            6
Threshold               3
Version                 n/a
Build Date              n/a
Storage Type            raft
HA Enabled              true
HA Cluster              n/a
HA Mode                 active
Active Since            2024-05-13T07:54:20.471072843Z
Active Node Address     <none>
Raft Committed Index    78
Raft Applied Index      78
```

2. Unix

Конфигурация слушателя Unix настраивает StarVault на прослушивание на указанном Unix доменном сокете.

```
listener "unix" {
  address = "/run/vault.sock"
```

```
}
```

Блок конфигурации listener может быть указана более одного раза, чтобы заставить StarVault прослушивать несколько сокетов.

2.1. Параметры слушателя Unix

address

Обязательный строковый параметр. Указывает адрес для привязки Unix сокета.

socket_mode

Необязательный строковый параметр. Изменяет права доступа и специальные флаги Unix сокета.

socket_user

Необязательный строковый параметр. Изменяет пользователя-владельца Unix сокета.

socket_group

Необязательный строковый параметр. Изменяет группу-владельца Unix сокета.

2.2. Примеры конфигурации секции listener unix

Пример 8. Прослушивание на нескольких сокетах

В следующем примере StarVault настроен на прослушивание на указанном сокете и сокете по умолчанию.

```
listener "unix" {}

listener "unix" {
  address = "/var/run/vault.sock"
}
```

Пример 9. Прослушивание на нескольких интерфейсах

В примере ниже StarVault настроен на прослушивание на указанном Unix сокете, а также на loopback интерфейсе.

```
listener "unix" {
  address = "/var/run/vault.sock"
}

listener "tcp" {
```

```
  address = "127.0.0.1:8200"
}
```

Пример 10. Настройка прав

В следующем примере показана конфигурация прав и владельца (пользователя и группы).

```
listener "unix" {
  address = "/var/run/vault.sock"
  socket_mode = "644"
  socket_user = "1000"
  socket_group = "1000"
}
```

Логирование выполненных запросов

StarVault можно настроить на логирование выполненных запросов с помощью параметра `log_requests_level`.

1. Активация логирования выполненных запросов

По умолчанию логирование завершенных запросов отключено. Чтобы активировать логирование запросов, установите параметр `log_requests_level` в конфигурации сервера StarVault на нужный уровень ведения журнала. Допустимые уровни логгирования: `error`, `warn`, `info`, `debug`, `trace` и `off`, используемый по умолчанию.

Завершенные запросы будут логироваться на настроенном уровне, если уровень логиов StarVault включает этот уровень логгирования. Например, если для `log_level` установлено значение `debug`, а для `log_requests_level` установлено значение `trace`, выполненные запросы не будут логироваться.

Если сервер StarVault уже запущен, то после настройки параметра в конфигурации сервера StarVault отправите процессу StarVault сигнал `SIGHUP`.

BASH

```
log_requests_level = "debug"
log_level = "debug"

listener "tcp"
{
  # ...
}
```

2. Деактивация логирования выполненных запросов

Для деактивации логирования завершенных запросов, удалите параметр `log_requests_level` из конфигурации сервера StarVault или установите его в значение `off`, и отправьте процессу StarVault сигнал `SIGHUP`.

BASH

```
log_requests_level = "off"
log_level = "debug"

listener "tcp"
{
```

```
  # ...
}
```