Кэширование. Постоянное кэширование

StarVault Proxy может восстанавливать токены и лизинги из файла постоянного кэша, созданного предыдущим процессом StarVault Proxy. Постоянный кэш — это файл BoltDB, который включает кортежи, зашифрованные сгенерированным ключом шифрования. Зашифрованные кортежи включают токен StarVault, используемый для извлечения секретов, аренды для токенов/секретов и секретные значения.

StarVault Proxy Persistent Caching восстановит только арендованные секреты. Секреты, которые не возобновляются, такие как KV v2, не будут сохранены.

Для использования постоянного кэша StarVault Proxy необходимо использовать автоматическую аутентификацию. Если токен автоматической аутентификации истек к моменту восстановления кэша, кэш будет аннулирован, и секреты необходимо будет повторно извлечь из StarVault.

Постоянный кэш StarVault Proxy в настоящее время поддерживается только в среде Kubernetes.

1. Типы постоянного кэша StarVault proxy

Информацию о доступных типах, их использовании и настройке см. на боковой панели.

2. Пример конфигурации постоянного кэша

Вот пример конфигурации постоянного кэша:

```
# Другие блоки конфигурации прокси-сервера StarVault
# ...

cache {
  persist "kubernetes" {
    path = "/vault/proxy-cache"
  }
}
```

2025 orionsoft. Все права защищены.

Программы

1. Инструменты HashiCorp

- **Агент хранилища** (StarVault Agent) может передавать информацию о секретах StarVault либо в файлы, либо непосредственно в дочерний процесс в виде переменных окружения, используя синтаксис шаблонов `consul-template`.
- **Прокси-хранилище** (StarVault Proxy) действует как API-прокси для StarVault и может по желанию разрешить или принудительно использовать автоматически аутентифицированный токен для взаимодействующих клиентов.
- **Провайдер хранилища Terraform** (Terraform Vault Provider) может читать, записывать и настраивать хранилище из HashiCorp Terraform.
- **Consul-template** - это шаблонный рендерер, нотификатор и супервизор для данных HashiCorp Consul и Vault.
- **Mault-ssh-helper** можно использовать для включения одноразовых паролей для аутентификации по SSH через StarVault.

2. Инструменты сторонних производителей

Следующий список инструментов поддерживается сообществом пользователей Vault; HashiCorp не проверяла и не утверждала их и не делает никаких заявлений об их пригодности или безопасности:

- Плагин HashiCorp Vault Jenkins - Плагин для Jenkins для добавления в среду сборки, засекреченную информацию.
- Spring Vault - проект Java Spring для работы с секретными данными Vault.
- pouch - набор инструментов для управления предоставлением секретов на хостах, основанных на методе аутентификации AppRole из Vault.
- goldfish - UI-панель StarVault, написанная с помощью VueJS и встроенного в StarVault Go API.
- vault-migrator - Инструмент для миграции данных между различными хранилищами StarVault.
- Cryptr - настольный пользовательский интерфейс Vault для Mac, Windows и Linux.
- sequelize-vault - Плагин Sequelize для простой интеграции секретов Vault.

- ansible-modules-hashivaul - модуль Ansible для настройки большинства элементов хранилища, включая ключи, бэкенды и политики.
- Docker credential helper - программа, которая автоматически считывает учетные данные с вашего сервера StarVault и передает их в Docker для аутентификации в вашем реестре при извлечении образа.
- Vault-CRD - Синхронизация ключей, хранящихся в HashiCorp Vault, с ключами Kubernetes Secrets для улучшения GitOps без ключей, хранящихся в файлах git manifest.
- vsh - Интерактивная оболочка с возможностью заполнения вкладок. Позволяет выполнять рекурсивные операции над адресами. Позволяет переносить ключи между обеими версиями KV.
- vault-cli - Инструмент автоматизации на основе YAML, который загружает кластер(ы) StarVault с нужной конфигурацией (пространства имен, конечные точки, политики, роли, конечные точки).
- vault-go - Помощник видов Golang Vault в качестве Kubernetes Custom Resource Definitions (CRD).
- HashiBox - среда Vagrant для имитации высокодоступного облака с Consul, Nomad, Vault и дополнительной поддержкой Waypoint. Поддерживается сообществом и предприятием.
- vkv - vkv - рекурсивный список записей ключевых значений из движка Vaults KV2 в различных форматах.

2025 orionsoft. Все права защищены.

Кэширование

StarVault Proxy Caching позволяет кэшировать на стороне клиента ответы, содержащие вновь созданные токены, и ответы, содержащие арендованные секреты, сгенерированные из этих вновь созданных токенов. Продление кэшированных токенов и аренд также управляется прокси.

1. Кэширование и обновления

Кэширование и обновления ответов управляются прокси-сервером только в этих конкретных сценариях:

1. Запросы на создание токенов выполняются через прокси-сервер. Это означает, что любые операции входа, выполняемые с использованием различных методов аутентификации и вызов конечных точек для создания токенов с использованием токен метода аутентификации через прокси-сервер, приведут к кэшированию ответа прокси-сервером. Ответы, содержащие новые токены, будут кэшироваться прокси-сервером только в том случае, если родительский токен уже управляется прокси-сервером или если новый токен является сиротским токеном.
2. Запросы на создание арендованных секретов выполняются через прокси-сервер с использованием токенов, которые уже управляются прокси-сервером. Это означает, что любые динамические учетные данные, выдаваемые с использованием токенов, управляемых прокси-сервером, будут кэшироваться, и их обновления будут учтены.

2. Постоянный кэш

StarVault Proxy может восстанавливать токены и договоры аренды из постоянного файла кэша, созданного предыдущим процессом StarVault Proxy.

Дополнительную информацию об этой функции см. на странице Постоянное кэширование StarVault Proxy.

3. Вытеснение кэша

Вытеснение записей кэша, относящихся к секретам, произойдет, когда прокси больше не сможет их обновлять. Это может произойти, когда секреты достигнут своего максимального TTL или если обновления приведут к ошибкам.

StarVault Proxy выполняет вытеснения кэша, отслеживая определенные типы запросов и коды ответов. Например, если запрос на отзыв токена сделан через прокси и если перенаправленный запрос на сервер StarVault успешен, то прокси вытесняет все записи кэша, связанные с отозванным токеном. Аналогично любая операция отзыва аренды также будет перехвачена прокси, и соответствующие записи кэша будут вытеснены.

Обратите внимание, что хотя прокси вытесняет записи кэша по истечении срока действия секрета и при перехвате запросов на отзыв, прокси все еще может быть полностью не осведомлен об отзывах, которые происходят через прямое взаимодействие клиента с сервером StarVault. Это может потенциально привести к устареванию записей кэша. Для управления устаревшими записями в кэше доступна конечная точка `/proxy/v1/cache-clear` (см. ниже), позволяющая вручную удалять записи кэша на основе некоторых критериев запроса, используемых для индексации записей кэша.

4. Запрос уникальности

Чтобы обнаружить повторные запросы и вернуть кэшированные ответы, прокси-сервер должен иметь способ уникальной идентификации запросов. Это вычисление в его нынешнем виде использует упрощенный подход (может измениться в будущем) сериализации и хеширования HTTP-запроса вместе со всеми заголовками и телом запроса. Это хэш-значение затем используется как индекс в кэше для проверки доступности ответа. Следствием этого подхода является то, что хэш-значение для любого запроса будет отличаться, если какие-либо данные в запросе изменены. Это имеет побочный эффект в виде ложных отрицательных результатов, если, скажем, изменен порядок параметров запроса. Пока запросы поступают без каких-либо изменений, поведение кэширования должно быть согласованным. Идентичные запросы с по-разному упорядоченными значениями запроса приведут к дублированию записей кэша. Предположение о том, что клиенты будут использовать согласованные механизмы для выполнения запросов, тем самым приводя к согласованным хэш-значениям на запрос, является идеей, на которой построена функциональность кэширования.

5. Управление обновлением

Токены и договоры аренды обновляются прокси-сервером с помощью секретного обновителя, который доступен через API Go сервера StarVault. Прокси-сервер выполняет все операции в памяти и ничего не сохраняет в хранилище. Это означает, что при отключении прокси-сервера все операции по обновлению немедленно прекращаются, и прокси-сервер не может возобновить обновления после этого. Обратите внимание, что отключение прокси-сервера не означает отзыва секретов, а означает лишь, что ответственность за обновление всех действительных неотозванных секретов больше не выполняется прокси-сервером StarVault.

6. API

6.1. Очистка кэша

Эта конечная точка очищает кэш на основе заданных критериев. Чтобы использовать этот API, необходимо заранее знать некоторую информацию о том, как прокси-сервер кэширует значения. Каждый ответ, кэшируемый в прокси-сервере, будет индексироваться по некоторым факторам в зависимости от типа запроса. Этими факторами могут быть `токен`, принадлежащий кэшированному ответу, `token_accessor` токена, принадлежащего кэшированному ответу, `request_path`, который привел к кэшированному ответу, `аренда`, прикрепленная к кэшированному ответу, `namespace`, к которому принадлежит кэшированный ответ, и еще несколько. Этот API раскрывает некоторые факторы, с помощью которых извлекаются и удаляются связанные записи кэша. Для слушателей без включенного кэширования этот API по-прежнему будет доступен, но ничего не будет делать (нет кэша для очистки) и вернет ответ `200`.

Пример вывода:

Метод	Путь	Вызывает
`POST`	`/proxy/v1/cache-clear`	`200 application/json`

Параметры

- `type (strings: required)` — тип записей кэша для вытеснения. Допустимые значения: `request_path`, `lease`, `toke n`, `token_accessor` и `all`. Если `тип` установлен на `all`, *очищается* весь кэш.
- `value (string: required)` — точное значение или префикс значения для выбранного `type`. Этот параметр необязателен, если `type` установлен на `all`.

6.2. Образец полезной нагрузки

C

```
{
  "type": "token",
  "value": "hvs.rlNjegSKykWcplOkwsjd8bP9"
}
```

6.3. Образец запроса

BASH

```
curl \
    --request POST \
```

```
    --data @payload.json \
    http://127.0.0.1:1234/proxy/v1/cache-clear
```

7. Конфигурация (cache)

Наличие блока `cache` верхнего уровня в любом виде (включая пустой блок `cache`) включит кэш. Блок `cache` верхнего уровня имеет следующую запись конфигурации:

- `persist (object: optional)` — конфигурация для постоянного кэша.

> Когда блок `cache` определен, в конфигурации также должен быть определен listener, в противном случае кэш использовать невозможно.

8. Конфигурация (persist)

Это общие значения конфигурации, которые находятся в `persist` блоке:

- `type (string: required)` — тип используемого постоянного кэша, например `kubernetes`.
- `path (string: required)` — путь на диске, по которому должен быть создан или восстановлен файл постоянного кэша.
- `keep_after_import (bool: Optional)` — если установлено значение `true`, восстановленный файл кэша не удаляется. По умолчанию — `false`.
- `exit_on_err (bool: Optional)` — если установлено значение true, если во время восстановления постоянного кэша возникнут какие-либо ошибки, StarVault Proxy завершит работу с ошибкой. По умолчанию — `true`.
- `service_account_token_file (string: Optional)` — если `type` установлено значение `kubernetes`, настраивает путь на диске, по которому можно найти токен учетной записи службы Kubernetes. По умолчанию используется `/var/run/secrets/kubernetes.io/serviceaccount/token`.

9. Конфигурация (listener)

- `listener (array of objects: required)` — конфигурация для слушателей.

На верхнем уровне может быть один или несколько блоков `listener`. Добавление слушателя включает API Proxy и позволяет API Proxy использовать кэш, если настроено. Эти значения конфигурации являются общими для блоков listener `tcp` и `unix`. Блоки типа `tcp`

поддерживают стандартные параметры слушателя `tcp`. Кроме того, параметр строки роли доступен как часть верхнего уровня блока `listener`, который можно настроить на `metrics_only` для обслуживания только метрик или на роль по умолчанию `default`, которая обслуживает все (включая метрики).

- `type (string: required)` — тип слушателя для использования. Допустимые значения: `tcp` и `unix`.

  при использовании HCL это может использоваться как ключ для блока, например, `listener "tcp" {…}`.

- `address (string: required)` — адрес, который прослушивает listener. Это может быть либо URL-путь при использовании `tcp`, либо путь к файлу при использовании `unix`. Например, `127.0.0.1:8200` или `/path/to/socket`. По умолчанию `127.0.0.1:8200`.

- `tls_disable (bool: false)` — указывает, будет ли отключен TLS.

- `tls_key_file (string: Optional)` — указывает путь к закрытому ключу для сертификата.

- `tls_cert_file (string: Optional)` — указывает путь к сертификату для TLS.

10. Пример конфигурации

Ниже приведен пример конфигурации кэша с дополнительным `persist` блоком, а также обычным слушателем и слушателем, который обслуживает только метрики.

```
# Другие блоки конфигурации прокси-сервера StarVault
# ...

cache {
    persist = {
        type = "kubernetes"
        path = "/vault/proxy-cache/"
        keep_after_import = true
        exit_on_err = true
        service_account_token_file = "/tmp/serviceaccount/token"
    }
}

listener "tcp" {
    address = "127.0.0.1:8100"
    tls_disable = true
}
```

```
listener "tcp" {
    address = "127.0.0.1:3000"
    tls_disable = true
    role = "metrics_only"
}
```

2025 orionsoft. Все права защищены.