Блок конфигурации Seal

Блок конфигурации Seal настраивает тип механизма запечатывания, который используется для дополнительной защиты данных. Например, это может быть решение Cloud KMS для шифрования и дешифрования корневого ключа. Этот блок конфигурации необязателен, и если не настроен, то StarVault будет использовать алгоритм Шамира для криптографического разделения корневого ключа.

1. Конфигурирование

Способ запечатывания можно настроить с помощью блока конфигурации Seal в файле конфигурации StarVault:

```
seal [NAME] {
  # ...
}
```

Например:

```
seal "transit" {
  # ...
}
```

Для вариантов конфигурации, которые также считывают переменную окружения, блок конфигурации Seal имеет приоритет над значениями в файле конфигурации.

2. Механизм запечатывания Transit

Механизм запечатывания Transit настраивает StarVault на использование Transit Secret Engine (модуль работы с транзитными секретами) в качестве механизма автоматического запечатывания. Механизм запечатывания Transit активируется одним из следующих способов:

- Блок `seal "transit"` в файле конфигурации StarVault
- Переменная окружения `VAULT_SEAL_TYPE` со значением `transit`.

2.1. Пример `transit`

В этом примере показана настройка механизма запечатывания Transit через указание всех требуемых значений в файле конфигурации StarVault:

```
seal "transit" {
  address            = "https://starvault:8200"
  token              = "s.Qf1s5zigZ4OX6akYjQXJC1jY"
  disable_renewal    = "false"

  // Key configuration
  key_name           = "transit_key_name"
  mount_path         = "transit/"

  // TLS Configuration
  tls_ca_cert        = "/etc/starvault/ca_cert.pem"
  tls_client_cert    = "/etc/starvault/client_cert.pem"
  tls_client_key     = "/etc/starvault/ca_cert.pem"
  tls_server_name    = "starvault"
  tls_skip_verify    = "false"
}
```

2.2. Параметры `transit`

Эти параметры применяются к блоку конфигурации seal в файле конфигурации StarVault:

- `address` (`string: <required>`): Полный адрес кластера StarVault. Также можно указать с помощью переменной окружения `VAULT_ADDR`.
- `token` (`string: <required>`): Токен StarVault, который будет использоваться. Его также можно указать с помощью переменной окружения `VAULT_TOKEN`.
- `key_name` (`string: <required>`): Транзитный ключ, который будет использоваться для шифрования и дешифрования. Его также можно указать с помощью переменной окружения `VAULT_TRANSIT_SEAL_KEY_NAME`.
- `mount_path` (`string: <required>`): Путь к точке монтирования модуля работы с транзитными секретами. Его также можно указать с помощью переменной окружения `VAULT_TRANSIT_SEAL_MOUNT_PATH`.
- `disable_renewal` (`string: "false"`): Параметр отключает автоматическое продление токена в случае, если жизненным циклом токена управляет какой-либо другой механизм вне StarVault, например StarVault Agent. Его также можно указать с помощью переменной окружения `VAULT_TRANSIT_SEAL_DISABLE_RENEWAL`.
- `tls_ca_cert` (`string: ""`): полный адрес кластера StarVault. Также указывается с помощью переменной окружения `VAULT_ADDR`.
- `token` (`string: <required>`): токен StarVault, который будет использоваться. Также указывается с помощью переменной окружения `VAULT_TOKEN`.

- `key_name` (`string: <required>`): транзитный ключ, который будет использоваться для шифрования и дешифрования. Также указывается с помощью переменной окружения `VAULT_TRANSIT_SEAL_KEY_NAME`.
- `mount_path` (`string: <required>`): путь к точке монтирования модуля работы с транзитными секретами. Также указывается с помощью переменной окружения `VAULT_TRANSIT_SEAL_MOUNT_PATH`.
- `disable_renewal` (`string: "false"`): параметр отключает автоматическое продление токена в случае, если жизненным циклом токена управляет какой-либо другой механизм вне StarVault, например StarVault Agent. Также указывается с помощью переменной окружения `VAULT_TRANSIT_SEAL_DISABLE_RENEWAL`.
- `tls_ca_cert` (`string: ""`): параметр указывает путь к файлу сертификата ЦС, используемого для связи с сервером StarVault. Также указывается с помощью переменной окружения `VAULT_CACERT`.
- `tls_client_cert` (`string: ""`): параметр указывает путь к сертификату клиента для связи с сервером StarVault. Также указывается с помощью переменной окружения `VAULT_CLIENT_CERT`.
- `tls_client_key` (`string: ""`): параметр указывает путь к закрытому ключу для связи с сервером StarVault. Также указывается с помощью переменной окружения `VAULT_CLIENT_KEY`.
- `tls_server_name` (`string: ""`): имя, используемое в качестве хоста SNI при подключении к серверу StarVault через TLS. Также указывается с помощью переменной окружения `VAULT_TLS_SERVER_NAME`.
- `tls_skip_verify` (`bool: "false"`): параметр отключает проверку сертификатов TLS. Использовать этот параметр крайне не рекомендуется, поскольку это снижает безопасность передачи данных на сервер StarVault и с него. Также указывается с помощью переменной окружения `STARVAULT_SKIP_VERIFY`.
- `disabled` (`string: ""`): установите значение параметра в `true`, если StarVault мигрирует с конфигурации, где было предусмотрено автоматическое запечатывание. В противном случае установите в значение `false`.

3. Аутентификация

Значения, связанные с аутентификацией, должны быть предоставлены либо как переменные окружения, либо как параметры конфигурации.

Хотя конфигурационный файл позволяет передавать `VAULT_TOKEN` как часть параметров печати, настоятельно рекомендуется задавать эти значения через переменные окружения.

Токен StarVault, используемый для аутентификации, должен иметь следующие разрешения на транзитном ключе:

```
path "<mount path>/encrypt/<key name>" {
  capabilities = ["update"]
}

path "<mount path>/decrypt/<key name>" {
  capabilities = ["update"]
}
```

Примечания для используемого токена:

- возможно должен быть свободный токен, в противном случае, когда срок действия родительского токена истечет или он будет отозван, механизм запечатывания перестанет работать.
- подумайте о том, чтобы сделать его периодическим токеном и не задавать явным образом максимальное время жизни (TTL), иначе в какой-то момент он перестанет быть возобновляемым.

4. Ротация ключей

Этот механизм запечатывания поддерживает ротацию ключей через соответствующие конечные точки в модуле работы с транзитными секретами. См. документацию. Старые ключи нельзя отключать или удалять – они используются для расшифровки старых данных.

Конфигурационная опция Service registration

Необязательный блок конфигурации `service_registration` настраивает имеющийся в StarVault механизм регистрации сервисов. Блок конфигурации `service_registration` предназначен для случаев, когда планируется использование системы для обнаружения сервисов типа Consul, но используете другую систему для бэкенда хранения.

Когда Consul настроен как бэкенд хранения, StarVault естественным образом использует Consul для регистрации сервисов, так что строка конфигурации `service_registration` не нужна.

Для случаев, когда хотите использовать другой бэкэнд хранения, например Raft, но при этом иметь возможность регистрации сервисов, можно использовать блок конфигурации `service_registration`:

```
service_registration "consul" {
  address = "127.0.0.1:8500"
}
storage "raft" {
  path = "/path/to/raft/data"
  node_id = "raft_node_1"
}
```

1. Конфигурирование блока `service_registration`

Настройка регистрации сервисов осуществляется в конфигурационном файле StarVault с помощью блока `service_registration`:

```
service_registration [NAME] {
  [PARAMETERS...]
}
```

Например

```
service_registration "consul" {
  address = "127.0.0.1:8500"
}
```

Для параметров конфигурации, которые также считывают переменную окружения, переменная окружения имеет приоритет над значениями в файле конфигурации.

2. Регистрация сервиса Consul

Consul Service Registration регистрирует StarVault как службу в Consul с проверкой работоспособности по умолчанию. Если Consul настроен как бэкенд хранилища, блок конфигурации `service_registration` не нужен, так как Consul автоматически зарегистрирует StarVault как сервис.

2.1. Конфигурирование сервиса Consul

```
service_registration "consul" {
  address      = "127.0.0.1:8500"
}
```

Если StarVault работает в режиме HA, включите в адрес протокол передачи данных (`http://` или `https://`):

```
service_registration "consul" {
  address      = "http://127.0.0.1:8500"
}
```

После правильной настройки, установка распечатанного StarVault должна быть доступна по:

TERMINAL

```
active.vault.service.consul
```

Инстанс распечатанного StarVault в режиме ожидания доступны по адресу:

TERMINAL

```
standby.vault.service.consul
```

Проверка состояния всех экземпляров распечатанного StarVault доступны по:

TERMINAL

```
vault.service.consul
```

Запечатанные экземпляры StarVault помечают себя как недостуаные, чтобы избежать возврата на уровне обнаружения сервисов Consul.

2.2. Параметры `consul`

- `address (string: "127.0.0.1:8500")` — адрес агента Consul, с которым необходимо установить связь. Это может быть IP-адрес, запись DNS или сокет unix. Рекомендуется взаимодействовать с локальным агентом Consul; не следует взаимодействовать с сервером напрямую.
- `check_timeout (string: "5s")` — интервал проверки, используемый для отправки информации о проверке состояния обратно в Consul. Он задается с помощью суффикса метки, например `"30s"` или `"1h"`.
- `disable_registration (string: "false")` — разрешение StarVault регистрировать себя в Consul.
- `scheme (string: "http")` — схема взаимодействии с Consul. Можно установить значение "http" или "https". Рекомендуется взаимодействовать с Consul по протоколу https при нелокальных соединениях. При общении через unix-сокет эта опция игнорируется.
- `service (string: "vault")` — имя сервиса для регистрации в Consul.
- `service_tags (string: "")` — список тегов, разделенных запятыми, которые необходимо прикрепить к регистрации сервиса в Consul.
- `service_address (string: nil)` — адрес службы, который необходимо установить во время регистрации службы в Consul. Если параметр не задан, StarVault будет использовать тот адрес перенаправления HA, который ему известен, что обычно желательно. Если установить этот параметр в `""`, Consul будет динамически использовать конфигурацию узла, на котором зарегистрирована служба. Это может быть полезно, если собираетесь использовать параметр Consul `translate_wan_addrs`.
- `token (string: "")` — токен Consul ACL с правами на регистрацию службы StarVault в каталоге служб Consul. Это **не** токен StarVault. Обратитесь за помощью к разделу ACL ниже.

Следующие настройки применяются при взаимодействии с Consul через зашифрованное соединение. Подробнее о шифровании соединений Consul можно прочитать на странице "Шифрование Consul".

- `tls_ca_file (string: "")` — путь к сертификату CA, для связи с Consul. Если путь не указан, по умолчанию используется системный пучок. Он должен быть установлен в соответствии с настройкой ca_file в Consul.
- `tls_cert_file (string: "")` (опционально) — путь к сертификату для связи с Consul. Он должен быть установлен в соответствии с настройкой `cert_file` в Consul.
- `tls_key_file (string: "")` — путь к приватному ключу, для связи с Consul. Он должен быть установлен в соответствии с настройкой `key_file` в Consul.
- `tls_min_version (string: "tls12")` — минимальная версия TLS для использования. Принимаются следующие значения: `"tls10"`, `"tls11"`, `"tls12"` or `"tls13"`.

- `tls_skip_verify` `(string: "false")` — отключение проверки сертификатов TLS. Не рекомендуется использовать эту опцию.

2.3. ACLs

Если используете ACL в Consul, для регистрации службы StarVault понадобятся соответствующие разрешения. Следующая политика ACL подойдет для большинства сценариев использования, предполагая, что имя службы `vault`:

```
{
  "service": {
    "vault": {
      "policy": "write"
    }
  }
}
```

2.4. Примеры `consul`

2.4.1. Локальный агент

В этом примере показан пример конфигурации, взаимодействующая с локальным агентом Consul, работающим по адресу `127.0.0.1:8500`.

```
service_registration "consul" {}
```

2.4.2. Детальная настройка

В этом примере показано взаимодействие с Consul по пользовательскому адресу с токеном ACL.

```
service_registration "consul" {
  address = "10.5.7.92:8194"
  token   = "abcd1234"
}
```

2.4.3. Взаимодействие Consul через сокет unix

В этом примере показано взаимодействие с Consul через локальный сокет unix.

```
service_registration "consul" {
  address = "unix:///tmp/.consul.http.sock"
}
```

2.4.4. Пользовательский TLS

В этом примере показано использование пользовательского CA, сертификата и файла ключа для безопасного взаимодействия с Consul по протоколу TLS.

```
service_registration "consul" {
  scheme        = "https"
  tls_ca_file   = "/etc/pem/vault.ca"
  tls_cert_file = "/etc/pem/vault.cert"
  tls_key_file  = "/etc/pem/vault.key"
}
```

3. Регистрация сервисов в Kubernetes

При регистрации сервисов в Kubernetes поды StarVault помечаются тегами, которые отражают их текущий статус и позволяют затем их использовать с селекторами. Регистрация сервисов возможна только тогда, когда StarVault работает в режиме высокой доступности.

3.1. Конфигурирование блока `service_registration` для регистрации сервисов в Kubernetes

```
service_registration "kubernetes" {}
```

Для успешной регистрации сервиса StarVault должен иметь возможность применять метки к подам в Kubernetes. Чтобы служебная учетная запись, ассоциированная с подами StarVault, могла обновлять спецификацию собственных подов, необходимы следующие правила управления доступом на основе ролей:

YAML

```yaml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: starvault
  name: starvault-discovery-role
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "update", "patch",”watch”,”list”]
```

3.2. Примеры включения регистрации сервисов в Kubernetes

После правильной настройки, включение регистрации сервисов приведет к тому, что поды Kubernetes получат следующие метки:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: starvault
  labels:
    vault-active: "false"
    vault-initialized: "true"
    vault-perf-standby: "false"
    vault-sealed: "false"
    vault-version: 1.2.0
```

После выключения поды StarVault будут иметь следующие метки:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: starvault
  labels:
    vault-active: "false"
    vault-initialized: "false"
    vault-perf-standby: "false"
    vault-sealed: "true"
    vault-version: 1.2.0
```

3.3. Описание меток подов

- `vault-active` (`string: "true"/"false"`) – Метка "StarVault active" обновляется динамически при каждой смене статуса активности StarVault. Значение "True" указывает, что этот под Vault ведущий. Значение "False" указывает на то, что этот под StarVault резервный.
- `vault-initialized` (`string: "true"/"false"`) – Метка "StarVault initialized" обновляется динамически при каждой смене статуса инициализации StarVault. Значение "True" указывает на то, что Vault инициализирован. Значение "False" указывает на то, что StarVault не инициализирован.
- `vault-perf-standby` (`string: "true"/"false"`) – Метка "StarVault performance standby" обновляется динамически при каждой смене статуса StarVault "ведущий/резервный". Это поле имеет значение только в том случае, если под является членом резервного рабочего кластера. В противном случае ему будет присвоено значение "false". Значение "True" указывает на то, что данный под StarVault — резервный рабочий под. Значение "False" указывает на то, что под StarVault — ведущий рабочий под.

- `vault-sealed` (`string: "true"/"false"`) – Метка "StarVault sealed" обновляется динамически при каждой смене статуса StarVault "запечатан/распечатан". Значение "True" указывает на то, что StarVault запечатан. Значение "False" указывает на то, что StarVault распечатан.
- `vault-version` (`string: "1.2.0"`) – Метка "StarVault version" – это строка, которая не меняется на протяжении жизненного цикла пода.

3.4. Пример сервиса

Присвоив поду метки, можно создавать сервисы с помощью селекторов, чтобы фильтровать поды с определенными ролями StarVault HA, тем самым обеспечивая прямую связь с подмножествами подов StarVault.
Обратите внимание на строку: `vault-active: "true"`.

YAML

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/instance: starvault
    app.kubernetes.io/name: starvault
    helm.sh/chart: starvault-1.2.0
  name: starvault-active
  namespace: starvault
spec:
  clusterIP: 10.7.254.51
  ports:
  - name: http
    port: 8200
    protocol: TCP
    targetPort: 8200
  - name: internal
    port: 8201
    protocol: TCP
    targetPort: 8201
  publishNotReadyAddresses: false
  selector:
    app.kubernetes.io/instance: starvault
    app.kubernetes.io/name: starvault
    component: server
    vault-active: "true"
  type: ClusterIP
```

Кроме того, задав `publishNotReadyAddresses: false`, можно удалить отказавшие поды из пула сервиса.

После создания, сервис из примера можно использовать как выделенную конечную точка, которая всегда будет достигать активного узла. При настройке репликации StarVault ее можно использовать в качестве основного адреса:

```bash
$ starvault write -f sys/replication/performance/primary/enable \
    primary_cluster_addr='https://starvault-
active.starvault.svc.cluster.local:8201'
```

3.5. Примеры обновлений

Метки подов в сочетании со стратегией обновления `OnDelete` сильно облегчают процесс управления обновлениями:

```bash
$ helm upgrade starvault oci://hub.orionsoft.ru/public/starvault --
set='server.image.tag=1.2.0' --reuse-values

$ kubectl delete pod --selector=vault-active=false \
    --selector=vault-version=1.2.0

$ kubectl delete pod --selector=vault-active=true \
    --selector=vault-version=1.2.0
```

При удалении экземпляра пода параметр `Statefulset`, определяющий желаемое состояние кластера, перепланирует удаленные поды, использовав для них последний образ.

Блок конфигурации Storage

Блок конфигурации `storage` настраивает бэкенд хранения, который представляет собой место для долговременного хранения информации StarVault. У каждого бэкенда есть свои плюсы, минусы, преимущества и компромиссы. Например, некоторые бэкенды поддерживают высокую доступность, в то время как другие предоставляют более надежный процесс резервного копирования и восстановления.

1. Конфигурация

Настройка бэкенда хранилища выполняется в конфигурационном файле StarVault с помощью секции `storage`:

```
storage [NAME] {
  [PARAMETERS...]
}
```

Например:

```
storage "file" {
  path = "/mnt/starvault/data"
}
```

Для параметров конфигурации, которые также считывают переменную окружения, переменная окружения будет иметь приоритет над значениями в файле конфигурации.

2. Поддерживаемые бэкенды хранения:

- Filesystem
- In-memory
- PostgreSQL
- Integrated Storage