Блок конфигурации telemetry

Блок конфигурации `telemetry` задает различные конфигурации для публикации метрик StarVault в вышестоящие системы. Доступные метрики StarVault можно найти в документации Telemetry internals.

```bash
telemetry {
  statsite_address = "statsite.company.local:8125"
}
```

1. Параметры `telemetry`

Из-за большого количества настраиваемых параметров в блоке `telemetry` параметры на этой странице сгруппированы по провайдерам телеметрии.

1.1. Общие параметры

Опции доступные для всех конфигураций телеметрии.

- `usage_gauge_period (string: "10m")` — интервал, через который собираются данные об использовании с высокой кардинальностью, такие как количество токенов, количество сущностей и количество секретов. Значение "none" отключает сбор данных. Используются формата строки время действия.
- `maximum_gauge_cardinality (int: 500)` — максимальная кардинальность калибровочных меток.
- `disable_hostname (bool: false)` — должны ли значения датчиков иметь префикс с именем локального хоста.
- `enable_hostname_label (bool: false)` — должны ли все значения метрики содержать метку `host` с локальным именем хоста. Рекомендуется включить `disable_hostname`, если используется эта опция.
- `lease_metrics_epsilon (string: "1h")` — размер контейнера, используемого для оценки истечения срока аренды в будущем. Например, для значения по умолчанию 1 час метрика `vault.expire.leases.by_expiration` будет суммировать общее количество истекающих аренд по 1-часовым контейнерам, начиная с текущего времени. Обратите внимание, что аренда распределяется по контейнерам путем округления. Например, если `lease_metrics_epsilon` имеет значение 1h и срок аренды A истекает через 25 минут, а срок аренды B - через 35 минут, то аренда A будет находиться в

первом контейнере, что соответствует 0-30 минутам, а аренда B - во втором контейнере, что соответствует 31-90 минутам. Используются формата строки время действия.

- `num_lease_metrics_buckets (int: 168)` — количество контейнеров с истекшим сроком действия для аренд. Например, для значения по умолчанию будет сообщено о 168 метках значений для метрики `vault.expire.leases.by_expiration`, где каждое значение в каждом контейнере отделено по времени параметром `lease_metrics_epsilon`. Для значения `lease_metrics_epsilon` по умолчанию 1 час и значения по умолчанию `num_lease_metrics_buckets`, `vault.expire.leases.by_expiration` будет сообщать общее количество аренд, истекающих в течение каждого часа от текущего времени до одной недели от текущего времени.
- `add_lease_metrics_namespace_labels (bool: false)` — если значение установлено в true, то `vault.expire.leases.by_expiration` будет разбивать истекающие аренды как по времени, так и по пространству имен. По умолчанию этот параметр отключен, так как его включение может привести к большой кардинальности метрики.
- `filter_default (bool: true)` — параметр определяет, разрешать ли метрики, которые не были указаны фильтром. По умолчанию установлено значение `true`, которое разрешает все метрики, если фильтры не указаны. При значении `false` без фильтров метрики отправляться не будут.
- `prefix_filter (string array: [])` — список правил фильтрации для разрешения/блокировки метрик по префиксам в следующем формате:

```bash
["+vault.token", "-vault.expire", "+vault.expire.num_leases"]
```

"+" пред префиксом разрешает любые метрики с данным префиксом. "-" перед префиксом — блокирует их. Если два правила пересекаются, приоритет будет иметь более конкретное правило. Блокировка будет иметь приоритет, если один и тот же префикс указан в списке несколько раз.

1.2. statsite

Параметры `telemetry` для statsite.

- `statsite_address (string: "")` — адрес сервера statsite для пересылки данных метрик.

```bash
telemetry {
  statsite_address = "statsite.company.local:8125"
}
```

1.3. statsd

Параметры `telemetry` для statsd.

- `statsd_address (string: "")` — адрес сервера statsd для пересылки метрик.

```
telemetry {
  statsd_address = "statsd.company.local:8125"
}
```

1.4. circonus

Параметры `telemetry` для Circonus.

- `circonus_api_token (string: "")` — действительный токен Circonus API, используемый для создания/управления проверкой. Если токен указан, то управление метрикой включено.
- `circonus_api_app (string: "nomad")` — действительное имя приложения, связанное с API-токеном.
- `circonus_api_url (string: "https://api.circonus.com/v2")` — базовый URL-адрес используемый для контакта с Circonus API.
- `circonus_submission_interval (string: "10s")` — интервал, через который метрики передаются в Circonus.
- `circonus_submission_url (string: "")` — поле `check.config.submission_url` объекта Check API из ранее созданной HTTPTRAP-проверки.
- `circonus_check_id (string: "")` — идентификатор проверки (не пакет проверки) из ранее созданной HTTPTRAP-проверки. Числовая часть поля `check._cid` в объекте Check API.
- `circonus_check_force_metric_activation (bool: false)` — следует ли принудительно активировать метрики, которые уже существуют и не активны в данный момент. Если управление проверками включено, то по умолчанию добавляются новые метрики по мере их появления. Если метрика уже существует в проверке, то она не будет активирована. Этот параметр переопределяет такое поведение.
- `circonus_check_instance_id (string: "<hostname>:<application>")` — уникальная идентификации метрик, поступающих от данного экземпляра. Может использоваться для поддержания непрерывности метрик для переходных или эфемерных экземпляров при их перемещении в пределах инфраструктуры. По умолчанию значение параметра установлено как "имя хоста":"имя приложения" (например, "host123:nomad").

- `circonus_check_search_tag (string: <service>:<application>)` — специальный тег, который в сочетании с идентификатором экземпляра помогает сузить результаты поиска, если не указан ни URL-адрес представления, ни Check ID. По умолчанию задается "сервис":"приложение" (например, "service:nomad").
- `circonus_check_display_name (string: "")` — имя присваивоенное проверке при создании. Имя отображается в списке Circonus UI Checks.
- `circonus_check_tags (string: "")` — список дополнительных тегов, которые следует добавить к проверке при создании. Теги отделяются друг от друга запятыми.
- `circonus_broker_id (string: "")` — идентификатор конкретного Circonus Broker, который будет использоваться при создании новой проверки. Числовая часть поля `broker._cid` в объекте Broker API. Если управление метрикой включено и не указаны ни Submission URL, ни Check ID, будет предпринята попытка поиска существующей проверки с использованием Instance ID и Search Tag. Если проверка не найдена, будет создана новая HTTPTRAP-проверка. По умолчанию выбирается случайный Enterprise Broker или Circonus Public Broker по умолчанию.
- `circonus_broker_select_tag (string: "")` — специальный тег, который будет использоваться для выбора Circonus Broker, если не указан Broker ID. Лучше всего использовать этот тег в качестве подсказки, какой брокер должен быть использован в зависимости от того, где запущен данный экземпляр (например, конкретное географическое местоположение или центр обработки данных, dc:sfo).

1.5. dogstatsd

Параметры `telemetry` для DogStatsD.

- `dogstatsd_addr (string: "")` — адрес экземпляра DogStatsD. DogStatsD - это разновидность протокола statsd, совместимая с протоколом statsd, с дополнительной возможностью оформлять метрики тегами и информацией о событиях. Если это предусмотрено, StarVault будет отправлять различные телеметрические данные этому экземпляру для агрегирования. Это можно использовать для сбора информации о времени выполнения.
- `dogstatsd_tags (string array: [])` — список глобальных тегов, которые будут добавляться во все телеметрические пакеты, отправляемые в DogStatsD. Список строк, где каждая строка выглядит как "my_tag_name:my_tag_value".

1.6. prometheus

Параметры `telemetry` для prometheus.

- `prometheus_retention_time (string: "24h")` — количество времени, в течение которого метрики Prometheus сохраняются в памяти. Если установить значение 0,

телеметрия Prometheus будет отключена.

- `disable_hostname (bool: false)` — рекомендуется также включить опцию `disable_hostname`, чтобы избежать префиксации метрик с именем хоста.

Конечная точка `/v1/sys/metrics` доступна только на активных узлах и автоматически отключается на ожидающих узлах. Возможно включить конечную точку `/v1/sys/metrics` на ожидающих узлах, разрешив неаутентифицированный доступ к метрикам.

StarVault не использует пути Prometheus по умолчанию, поэтому Prometheus должен быть сконфигурирован указанным ниже путем. Обратите внимание, что использование `?format=prometheus` в пути не сработает, так как "?" будет пропущен, поэтому его необходимо указать в качестве параметра.

Токен StarVault требуется при `capabilities = ["read", "list"]` для /v1/sys/metrics. Опции Prometheus `bearer_token` или `bearer_token_file` должны быть добавлены в задание на очистку.

Пример блока job_name, необходимой в конфигурации Prometheus, приведен ниже.

YAML

```yaml
# prometheus.yml
scrape_configs:
  - job_name: 'starvault'
    metrics_path: "/v1/sys/metrics"
    params:
      format: ['prometheus']
    scheme: https
    tls_config:
      ca_file: your_ca_here.pem ①
    bearer_token: "your_vault_token_here"
    static_configs:
    - targets: ['your_vault_server_here:8200']
```

1. В качестве теста можно заменить на `insecure_skip_verify: true`

Ниже показан пример конфигурации телеметрии, которую нужно добавить в файл конфигурации StarVault:

BASH

```bash
telemetry {
  prometheus_retention_time = "30s"
  disable_hostname = true
}
```

1.7. stackdriver

Параметры `telemetry` для Stackdriver Monitoring.

Провайдер телеметрии Stackdriver использует официальный Google Cloud Golang SDK. Это означает, что он поддерживает распространенные способы предоставления учетных данных в Google Cloud.

```
https://www.googleapis.com/auth/cloud-platform
https://www.googleapis.com/auth/monitoring
https://www.googleapis.com/auth/monitoring.write
```

И следующие роли IAM:

```
roles/monitoring.metricWriter
```

- `stackdriver_project_id (string: "")` — Google Cloud ProjectID для отправки данных телеметрии.
- `stackdriver_namespace (string: "")` — идентификатор пространства имен для данных телеметрии.
- `stackdriver_debug_logs (bool: "false")` — записывать ли StarVault дополнительные debug логи, связанные с стекдрайвером, в стандартный вывод ошибок (stderr).

Рекомендуется также включить опцию `disable_hostname`, чтобы избежать префиксации метрик с именем хоста, и включить вместо нее опцию `enable_hostname_label`.

```
telemetry {
  stackdriver_project_id = "my-test-project"
  stackdriver_location = "us-east1-a"
  stackdriver_namespace = "vault-cluster-a"
  disable_hostname = true
  enable_hostname_label = true
}
```

Метрики из StarVault можно найти в Metrics Explorer. Все эти метрики отображаются с типом ресурса g`eneric_task`, а название метрики имеет префикс `custom.googleapis.com/go-metrics/`.

Блок конфигурации ui

StarVault имеет пользовательский интерфейс (веб-интерфейс) для взаимодействия с оператором. С помощью пользовательского интерфейса StarVault можно легко создавать, читать, обновлять и удалять секреты, проходить аутентификацию, распечатывать хранилище и многое другое.

1. Активация ui

По умолчанию пользовательский интерфейс StarVault не активирован. Чтобы активировать пользовательский интерфейс, установите параметр `ui` в конфигурации сервера StarVault в значение `true`.

```
ui = true

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

2. Доступ к StarVault UI

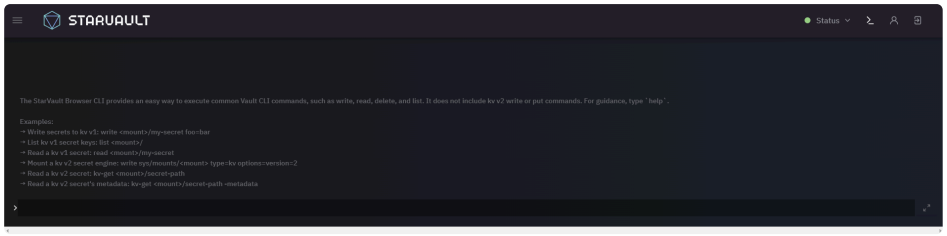
Пользовательский интерфейс работает на том же порту, что и слушатель StarVault. Поэтому для доступа к пользовательскому интерфейсу необходимо настроить хотя бы одну секцию `listener`.

```
listener "tcp" {
  address = "10.0.1.35:8200"
  ...
}
```

В этом случае пользовательский интерфейс доступен по URL-адресу `https://10.0.1.35:8200/ui/` с любой машины в подсети (при условии отсутствия или соответствующей настройке межсетевых экранов). Он также доступен по любой записи DNS, которая разрешается на этот IP-адрес.

При использовании TLS (рекомендуется) сертификат должен быть действителен для всех записей DNS, через которые вы будете обращаться к пользовательскому интерфейсу StarVault, и для всех IP-адресов в SAN. Если вы используете StarVault с самоподписанным сертификатом, в браузерах, которые обращаются к пользовательскому интерфейсу StarVault, должен быть установлен корневой ЦС. В противном случае браузер может вывести предупреждение о том, что сайт является "недоверенным". Настоятельно рекомендуется, чтобы клиентские браузеры, обращающиеся к пользовательскому интерфейсу StarVault, устанавливали соответствующий корневой ЦС для проверки, чтобы снизить вероятность атаки MITM.

Vault UI включает в себя интерактивный Web REPL для взаимодействия с API StarVault, подобно StarVault CLI.



Механизм секретов базы данных MySQL/MariaDB

Этот механизм может использовать внешние сертификаты X.509 как часть проверки TLS или подписи. Проверка подписей по сертификатам X.509, использующим SHA-1, является устаревшей и не используется без обходного пути. Дополнительную информацию см. в разделе FAQ об устаревании.

MySQL - один из поддерживаемых плагинов для механизма секретов баз данных. Этот плагин генерирует учетные данные базы данных динамически на основе настроенных ролей для базы данных MySQL, а также поддерживает статические роли.

Этот плагин имеет несколько различных экземпляров, встроенных в starvault, каждый из которых предназначен для разных драйверов MySQL. Единственное различие между этими плагинами заключается в длине имен пользователей, генерируемых плагином, так как разные версии mysql принимают разные длины. Доступны следующие плагины:

- mysql-database-plugin
- mysql-aurora-database-plugin
- mysql-rds-database-plugin
- mysql-legacy-database-plugin

Дополнительные сведения о настройке механизма секретов базы данных см. в документации по механизму секретов базы данных.

1. Возможности

Название плагина	Возможность ротации root	Динамические роли	Кастомизация имен пользователей
Зависит от ситуации (см.: выше)	Да	Да	Да

2. Настройка

1. Включите механизм секретов базы данных, если он еще не включен:

```
$ starvault secrets enable database
Success! Enabled the database secrets engine at: database/
```

По умолчанию движок secrets будет включаться по имени движка. Чтобы включить движок secrets по другому пути, используйте аргумент `-path`.

2. Настройте StarVault с помощью соответствующего плагина и информации о подключении:

```
$ starvault write database/config/my-mysql-database \
    plugin_name=mysql-database-plugin \
    connection_url="{{username}}:{{password}}@tcp(127.0.0.1:3306)/" \
    allowed_roles="my-role" \
    username="vaultuser" \
    password="vaultpass"
```

3. Настройте роль, которая сопоставляет имя в StarVault с оператором SQL, чтобы выполнить создание учетной записи базы данных:

```
$ starvault write database/roles/my-role \
    db_name=my-mysql-database \
    creation_statements="CREATE USER '{{name}}'@'%' IDENTIFIED BY
'{{password}}';GRANT SELECT ON *.* TO '{{name}}'@'%';" \
    default_ttl="1h" \
    max_ttl="24h"
Success! Data written to: database/roles/my-role
```

3. Использование

После того как механизм секретов настроен и у пользователя/машины есть токен StarVault с соответствующими правами, он может генерировать учетные данные.

1. Сгенерируйте новые учетные данные, считав из конечной точки `/creds` имя роли:

```
$ starvault read database/creds/my-role
```

Key	Value
`lease_id`	`database/creds/my-role/2f6a614c-4aa2-7b19-24b9-ad944a8d4de6`
`lease_duration`	`1h`

`lease_renewable`	`true`
`password`	`yY-57n3X5UQhxnmFRP3f`
`username`	`v_vaultuser_my-role_crBWVqVh2Hc1`

4. Проверка подлинности сертификата клиента x509

Этот плагин поддерживает использование аутентификации MySQL по сертификату x509 на стороне клиента

Чтобы использовать этот механизм аутентификации, настройте плагин:

```
$ starvault write database/config/my-mysql-database \
    plugin_name=mysql-database-plugin \
    allowed_roles="my-role" \
    connection_url="user:password@tcp(localhost:3306)/test" \
    tls_certificate_key=@/path/to/client.pem \
    tls_ca=@/path/to/client.ca
```

BASH

Параметры `tls_certificate_key` и `tls_ca` соответствуют параметрам `ssl-cert` (в сочетании с ssl-key) и `ssl-ca` из MySQL, за исключением того, что параметры StarVault - это содержимое этих файлов, а не имена файлов. Таким образом, эти две опции не зависят друг от друга. Дополнительные сведения см. в разделе Параметры подключения MySQL.

5. Примеры

5.1. Использование подстановочных символов в запросах о предоставлении прав доступа

MySQL поддерживает использование подстановочных символов для предоставления прав. Иногда это необходимо приложениям, которые ожидают доступа к большому количеству баз данных внутри MySQL. Это можно реализовать, используя подстановочный символ в операторе предоставления. Например, если вы хотите, чтобы пользователь, созданный в StarVault, имел доступ ко всем базам данных, начинающимся с `fooapp_`, вы можете использовать следующий оператор создания:

```
CREATE USER '{{name}}'@'%' IDENTIFIED BY '{{password}}'; GRANT SELECT ON
`fooapp\_%`.* TO '{{name}}'@'%';
```

BASH

MySQL ожидает, что часть, в которой должны быть помещены подстановочные символы, будет находиться внутри обратных знаков. Если вы хотите добавить этот оператор создания в StarVault через StarVault CLI, вы не можете просто вставить вышеприведенный оператор в CLI, потому что оболочка интерпретирует текст между знаками обратных кавычек как нечто, что должно быть выполнено. Самый простой способ обойти это - закодировать заявление о создании в Base64 и передать его в StarVault. Например:

```bash
$ starvault write database/roles/my-role \
    db_name=my-mysql-database \

creation_statements="Q1JFQVRFIFVTRVIgJ3t7bmFtZX19J0AnJScgSURFTlRJRklFRCBCWSAne3t
wYXNzd29yZH19JzsgR1JBTlQgU0VMRUNUIE9OIGBmb29hcHBcXyVgLiogVE8gJ3t7bmFtZX19J0AnJSc
7" \
    default_ttl="1h" \
    max_ttl="24h"
```

5.2. Ротация учетных данных root в MySQL 5.6

По умолчанию для ротации `root` в MySQL используется синтаксис `ALTER USER`, присутствующий в MySQL 5.7 и выше. Для MySQL 5.6 операторы ротации `root` должны быть настроены на использование старого синтаксиса `SET PASSWORD`. Например:

```bash
$ starvault write database/config/my-mysql-database \
    plugin_name=mysql-database-plugin \
    connection_url="{{username}}:{{password}}@tcp(127.0.0.1:3306)/" \
    root_rotation_statements="SET PASSWORD = PASSWORD('{{password}}')" \
    allowed_roles="my-role" \
    username="root" \
    password="mysql"
```

Руководство по ротации `root` учетных данных см. в разделе Ротация `root` учетных данных базы данных.

6. API

Полный список настраиваемых параметров можно посмотреть на странице API плагина базы данных MySQL.

Более подробную информацию о HTTP API движка секретов баз данных можно найти на странице API движка секретов баз данных.