Создание сертификатов для *Ingress* объектов

1. Цель

Цель статьи — создание сертификатов для *Ingress* объектов, включая этапы:

1. Создание ресурса *Issuer* типа `SelfSigned` в cert-manager;
2. Создание ресурса *Ingress* с аннотацией к созданному *Issuer*, который будет автоматически выпускать сертификаты для *Ingress*.

2. Cert-manager

cert-manager создает сертификаты TLS для рабочих нагрузок в кластере Kubernetes или в платформе Nova и обновляет сертификаты до истечения срока их действия.

cert-manager может получать сертификаты от различных центров сертификации, включая: StarVault, Let's Encrypt, HashiCorp Vault, Venafi и частные PKI.

С помощью ресурса сертификатов cert-manager'а, закрытый ключ и сертификат хранятся в Kubernetes Secret, который монтируется приложением Pod или используется контроллером *Ingress*. При использовании csi-driver, csi-driver-spiffe или istio-csr закрытый ключ генерируется по требованию, перед запуском приложения. Закрытый ключ никогда не покидает узел и не хранится в Kubernetes Secret.

3. Установка

cert-manager уже предустановлен в платформу Nova в пространство имен `nova-cert-managment`

Можете проверить установленный компонент командой:

```bash
$ kubectl get pods --namespace nova-cert-management
NAME                                          READY   STATUS    RESTARTS
AGE
nova-cert-manager-cainjector-759585c78d-tpwpn   1/1     Running   2 (15h ago)
26d
nova-cert-manager-dbc656674-g7c9b               1/1     Running   2 (15h ago)
26d
```

```
nova-cert-manager-webhook-6bfb9db944-ctxj6      1/1     Running   2 (15h ago)
26d
nova-trust-manager-65854d6d6c-mlgzj             1/1     Running   3 (15h ago)
26d
```

4. Issuers

`Issuers` и `ClusterIssuers` — это ресурсы Kubernetes, представляющие `центры сертификации (ЦС)`, которые могут генерировать подписанные сертификаты, выполняя запросы на подписание сертификатов. Все сертификаты cert-manager требуют наличия аннотации к `Issuer`.

5. StarVault Issuer

`StarVault` `Issuer` представляет центр сертификации StarVault — многоцелевое хранилище секретов, которое можно использовать для подписания сертификатов для вашей инфраструктуры открытых ключей (PKI). StarVault является внешним проектом для cert-manager, поэтому в данном руководстве предполагается, что он развернут правильно, настроен и готов к подписанию. Подробнее о том, как настроить StarVault в качестве центра сертификации, вы можете прочитать здесь.

Тип `Issuer` обычно используется, когда StarVault уже используется в вашей инфраструктуре.

Для импортирования собственного сертификата в качестве центра сертификации в StarVault можно воспользоваться следующей командой:

```bash
$ starvault write pki/config/ca pem_bundle=@ca_bundle.pem ①
```

1. ca_bundle.pem — файл, который последовательно содержит сертификат и ключ этого серитфиката.

5.1. Развертывание

Все `StarVault` `Issuer` имеют общую конфигурацию для запроса сертификатов: сервер, путь и сертификат ЦС:

- `Server` — URL, по которому можно получить доступ к StarVault.
- `Path` — путь к StarVault, который будет использоваться для подписи. Обратите внимание, что путь должен использовать конечную точку sign.

- `CA bundle` — необязательное поле, содержащее закодированную в base64 строку сертификата центра сертификации, которому следует доверять соединение со StarVault. Обычно это поле всегда требуется при использовании https URL.

Сертификат можно получить с виртуальной машины с установленным StarVault по пути `/opt/starvault/tls/tls.crt`. Ниже приведен пример части конфигурации с подключением сервера StarVault.

```yaml
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: starvault-issuer
  namespace: sandbox
spec:
  vault:
    path: pki_int/sign/example-dot-com
    server: https://starvault.vlab.local:8200
    caBundle: <base64 encoded CA Bundle PEM file>
    auth:
      ...
```

5.2. Аутентификация

5.2.1. A. Аутентификация через AppRole

AppRole — метод аутентификации в StarVault с помощью внутренней системы ролевой политики. Этот метод аутентификации требует, чтобы `Issuer` владел секретным ключом `SecretID`, `RoleID` роли, которую нужно принять, и путем к роли приложения. Секретный ключ ID должен храниться в Kubernetes `Secret`, который находится в том же пространстве имен, что и `Issuer`, или в пространстве имен ресурсов кластера в случае `ClusterIssuer`.

```yaml
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: cert-manager-vault-approle
  namespace: sandbox
data:
  secretId: "MDI..."
```

После создания секрета, `Issuer` готов к развертыванию, `Issuer` ссылается на этот секрет, а также на ключ данных поля, хранящего идентификатор секрета.

```yaml
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
```

```
  name: starvault-issuer
  namespace: sandbox
spec:
  vault:
    path: pki_int/sign/example-dot-com
    server: https://starvault.vlab.local:8200
    caBundle: <base64 encoded caBundle PEM file>
    auth:
      appRole:
        path: approle
        roleId: "291b9d21-8ff5-..."
        secretRef:
          name: cert-manager-vault-approle
          key: secretId
```

5.2.2. Б. Аутентификация с помощью учетных записей служб Kubernetes

Для метода Kubernetes Auth требуется `token_reviewer_jwt`, который будет использоваться StarVault для аутентификации на сервере Kubernetes API.

`token_reviewer_jwt` может быть долгоживущим токеном учетной записи службы. Убедитесь, что токен связан с учетной записью сервиса, которая имеет необходимые разрешения для вызова TokenReview API.

Создание учетной записи службы Kubernetes и долгоживущего токена:

BASH

```
$ kubectl create serviceaccount -n sandbox starvault-issuer
```

Манифесты StarVault будут выглядеть следующим образом:

YML

```
apiVersion: v1
kind: Secret
metadata:
  name: starvault-issuer-secret
  namespace: sandbox
  annotations:
    kubernetes.io/service-account.name: starvault-issuer
type: kubernetes.io/service-account-token
```

Разрешение учетной записи вызывать TokenReview API:

YML

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: starvault-auth-delegator
roleRef:
  apiGroup: rbac.authorization.k8s.io
```

```
  kind: ClusterRole
  name: system:auth-delegator
subjects:
- kind: ServiceAccount
  name: starvault-issuer
  namespace: sandbox
```

Получение `token_reviewer_jwt`:

BASH

```
$ kubectl get secret starvault-issuer-secret -n sandbox -o
jsonpath='{.data.token}' | base64 --decode -w0
```

Включение метода kubernetes auth в StarVault.

BASH

```
starvault auth enable -path=kubernetes-cluster001 kubernetes
kubectl config view --minify --flatten -ojson \
  | jq -r '.clusters[].cluster."certificate-authority-data"' \
  | base64 -d >/tmp/cacrt
starvault write auth/kubernetes-cluster001/config \
    token_reviewer_jwt="<Token>"
    kubernetes_host=<kubernetes-api-server-url> \
    kubernetes_ca_cert=@/tmp/cacrt
```

Добавление роли RBAC, чтобы cert-manager мог получать токены для ServiceAccount:

YML

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: starvault-issuer
  namespace: sandbox
rules:
  - apiGroups: ['']
    resources: ['serviceaccounts/token']
    resourceNames: ['starvault-issuer']
    verbs: ['create']
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: starvault-issuer
  namespace: sandbox
subjects:
  - kind: ServiceAccount
    name: nova-cert-manager
    namespace: nova-cert-management
roleRef:
  apiGroup: rbac.authorization.k8s.io
```

```
  kind: Role
  name: starvault-issuer
```

Создание `policy` с названием *policy-issuer* в StarVault с минимальными правами для подписи сертификата:

```bash
path "pki_int/sign/example-dot-com" {
     capabilities = ["create", "update"]}
```

Создание роль StarVault:

```bash
starvault write auth/kubernetes-cluster001/role/starvault-issuer-role \
    bound_service_account_names=starvault-issuer \
    bound_service_account_namespaces=sandbox \
    audience="vault://sandbox/starvault-issuer" \
    policies=policy-issuer \
    ttl=24h
```

Рекомендуется использовать разные роли StarVault для каждого `Issuer` или `ClusterIssuer`. `Audience` позволяет ограничить роль StarVault одним `Issuer` или `ClusterIssuer`. Синтаксис следующий:

```bash
"vault://<namespace>/<issuer-name>" ①
"vault://<cluster-issuer-name>" ②
```

1. Для Issuer.
2. Для ClusterIssuer.

Наконец, вы можете создать свой `Issuer`:

```yml
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: starvault-issuer
  namespace: sandbox
spec:
  vault:
    path: pki_int/sign/example-dot-com
    server: https://starvault.vlab.local:8200
    auth:
      kubernetes:
        role: starvault-issuer-role
        mountPath: /v1/auth/kubernetes-cluster001
        serviceAccountRef:
          name: starvault-issuer
```

5.3. Проверка развертывания `Issuer`

После развертывания `StarVault` `Issuer` будет отмечен как готовый, если его конфигурация соответствует действительности. Замените issuers ниже на clusterissuers, если развернут кластер.

`StarVault` `Issuer` проверяет StarVault, запрашивая конечную точку `v1/sys/health`, чтобы убедиться, что StarVault распечатан и инициализирован, прежде чем запрашивать сертификаты. Результат этого запроса заполнит столбец STATUS.

```bash
$ kubectl get issuers starvault-issuer -n sandbox -o wide

NAME               READY   STATUS           AGE
starvault-issuer   True    Vault verified   8s
```

Теперь сертификаты можно запрашивать с помощью `StarVault` `Issuer` с именем starvault-issuer в пространстве имен sandbox.

6. Аннотация к `Ingress` ресурсу

Частым случаем использования cert-manager является запрос сертификатов, подписанных TLS, для защиты `Ingress` ресурсов. Это можно сделать, просто добавив аннотации к ресурсам `Ingress`, и cert-manager облегчит создание ресурса сертификата. За это отвечает небольшой подкомпонент cert-manager, *ingress-shim*.

6.1. Как это работает

Подкомпонент *ingress-shim* следит за ресурсами `Ingress` в кластере. Если он обнаружит `Ingress` с аннотациями, описанными в разделе Поддерживаемые аннотации, то проверит, что в том же пространстве имен, где находится ресурс `Ingress`, существует ресурс `Certificate` с именем, указанным в поле `tls.secretName` и настроенным, как описано в этом `Ingress`.

Приведенный ниже манифест создаст `Ingress` ресурс с привязкой к ранее созданному `Issuer` `my-ca-issuer`:

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations: ①
    cert-manager.io/issuer: starvault-issuer
    cert-manager.io/common-name: www.example.com
  name: my-ingress
  namespace: sandbox
```

```
spec:
  rules:
  - host: www.example.com
    http:
      paths:
      - pathType: Prefix
        path: /
        backend:
          service:
            name: myservice
            port:
              number: 80
  tls: ②
  - hosts:
    - www.example.com
    secretName: myingress-cert ③
```

1. Добавление аннотации, указывающей какой issuer использовать
2. Размещение хоста в конфигурации TLS будет определять, что будет содержаться в subjectAltNames сертификата
3. cert-manager будет хранить сертификат в этом secret store

В платформе Nova, чтобы подписать сертификаты для своих сервисов достаточно указать в аннотации существующий ClusterIssuer `nova-dynamic-internal-cluster-issuer`.

6.2. Поддерживаемые аннотации

Можете указать следующие аннотации для ресурсов `Ingress`, чтобы вызвать автоматическое создание ресурсов сертификатов:

- `cert-manager.io/issuer`: имя `Issuer`, который должен выпустить сертификат, необходимый для данного `Ingress`

⚠️ Эта аннотация *не* предполагает наличие `Issuer` с привязкой к пространству имен. По умолчанию будет использоваться cert-manager.io `Issuer`, однако в случае внешних типов *Issuers* должно быть использовано как для типов *Issuers* с привязкой к пространству имен, так и для типов *Issuers* с привязкой ко всему кластеру.

⚠️ Если используется `Issuer` для пространства имен, то он должен находиться в том же пространстве имен, что и ресурс `Ingress`.

- `cert-manager.io/cluster-issuer`: имя ClusterIssuer для получения сертификата, необходимого для этого `Ingress`. Не имеет значения, в каком пространстве имен находится ваш `Ingress`, так как ClusterIssuers — это ресурс с привязкой ко всему кластеру.

⚠️ Эта аннотация является сокращением для ссылки на cert-manager.io ClusterIssuer без необходимости указывать группу и тип. Она не предназначена для указания внешнего `Issuer` с кластерной привязкой — для этого используйте аннотацию cert-manager.io/issuer.

Другие поддерживаемые опциональные аннотации вы можете найти на оффициальном сайте.

6.3. Генерация нескольких сертификатов с несколькими `Ingress`

Если нужно сгенерировать сертификаты из нескольких `Ingress`, убедитесь, что они имеют аннотацию `Issuer`. Помимо аннотации, необходимо, чтобы каждый `Ingress` обладал уникальным именем `tls.secretName`

6.4. Устранение неполадок

Если после применения аннотаций *ingress-shim* ресурс `Certificate` не создается, проверьте, установлена ли хотя бы одна аннотация `cert-manager.io/issuer` или `cert-manager.io/cluster-issuer`.

Платформа управления безопасностью контейнеров Neuvector

В данном разделе документации вы можете получить подробную информацию по платформе Neuvector, предназначенной для управления безопасностью контейнеров в кластерах Nova Container Platform.

Платформа NeuVector является одним из дополнительных модулей Nova Container Platform и устанавливается в кластер Kubernetes с целью обеспечить дополнительную безопасность и защиту контейнеров во время их выполнения.

Neuvector выпускается компанией SUSE, исходный код платформы открыт, доступен по лицензии Apache 2.0 в репозитории Github.

1. Основные возможности Neuvector в Nova Container Platform

Neuvector расширяет стандартные функции безопасности в Nova Container Platform и Kubernetes, а именно:

- Обеспечивает Nova Container Platform комплексным автоматизированным решением для защиты контейнеров во время их работы
- Обеспечивает среду Kubernetes необходимыми вебхуками для контроля запросов к серверу Kubernetes API
- Предоставляет в интерактивном графическом интерфейсе полную картину движения сетевого трафика (как для E-W, так и N-S)
- Использует поведенческий анализ сервисов, запущенных в контейнерах и автоматическую изоляцию нелегитимных сервисов
- Использует L7 межсетевой экран для блокировки несанкционированных соединений между контейнерами
- Использует механизмы автоматического обнаружения и предотвращения атак на контейнеры
- Предоставляет механизмы обеспечения безопасности всей цепочки процессов CI/CD, а именно:
    - Сканирование образов контейнеров на уязвимости
    - Сканирование хранилищ образов контейнеров на уязвимости

    - Проверка цифровой подписи образов контейнеров
    - Интерфейс для запуска сканирования образа после его сборки из CI-пайплайнов
- Выполняет непрерывный аудит безопасности узлов Nova Container Platform и запущенных контейнеров
- Предоставляет инструментарий для работы с отчетными данными для оценки рисков
- Предоставляет инструментарий для экспорта событий безопасности в различные системы

2. Интеграция с Nova Container Platform

Модуль Neuvector в Nova Container Platform поставляется преднастроенным и содержит следующие интеграции, доступные сразу после его установки:

- Интеграция со службой непрерывного развертывания FluxCD, с помощью которой выполняется установка, настройка и поддержание консистентности модуля в кластере Kubernetes
- Интеграция с Nova OAuth: после установки модуля администратор кластера может выполнить вход в Neuvector с помощью OIDC и существующих учетных записей
- Интеграция с StarVault: конфигурационные файлы Neuvector, содержащие чувствительные данные, не хранятся в Kubernetes, а генерируются “на лету” из секретов в StarVault.
- Преднастроенные правила Admission Control, определяющие базовые политики контроля операций в Kubernetes
- Преднастроенные политики автоматического сканирования узлов и кластера Kubernetes
- Дополнительный сервис Neuvector API Docs с офлайн-документацией по работе с Neuvector API

3. Содержание раздела

Управление сертификатами

Данный раздел содержит статьи описывающие управление сертификатами в Nova Container Platform.

1. Глоссарий

Глоссарий содержит описание основных терминов, устоявшихся выражений, примитивов и определений, которые вы можете встретить в данной документации и при работе с инфраструктурой PKI в Nova Container Platform.

1.1. Инфраструктура открытых ключей

PKI (Public Key Infrastructure)

Инфраструктура открытых ключей является набором технических средств, а также распределённых служб и компонентов, в совокупности используемых для поддержки задач шифрования на основе закрытого и открытого ключей.

1.2. Центр сертификации

CA (Certification Authority)

Центр сертификации (удостоверяющий центр) является компонентом инфраструктуры PKI, который выдает цифровые сертификаты, осуществляет их подпись своим открытым ключом и хранит в базе данных сертификатов.

1.3. X.509

X.509 определяет стандартные форматы данных и процедуры распределения открытых ключей с помощью соответствующих сертификатов с цифровыми подписями.

2. Содержание раздела

- Обновление сертификатов
- Управление цепочками сертификатов