Проверка срока действия сертификатов платформы

1. Проверка сертификатов, выпущенных в StarVault

Вы можете проверить срок действия сертификатов платформы, выпущенных в StarVault, с помощью утилиты `nova-ctl`. Для этого выполните следующую команду:

```bash
nova-ctl certs check-expiration --ssh-key key.pem --ssh-user ec2-user
```

> В качестве аргументов `--ssh-key` и `--ssh-user` укажите информацию, использованную на этапе конфигурации ключевой пары SSH.

Пример

```bash
nova-ctl certs check-expiration --ssh-key key.pem --ssh-user ec2-user
```

```
┌──────────────────────┬──────────────────────────────┬──────────────────────┬──────────────┬──────────────────────┐
│ Node                 │ Certificate                  │ Expires              │ Residual time │ Certificate authority │
├──────────────────────┼──────────────────────────────┼──────────────────────┼──────────────┼──────────────────────┤
│ node-master-hnf8g804 │ etcd/etcd-admin              │ 2024-08-09 08:31:01 │ 365d         │ etcd-ca-int          │
│ node-master-hnf8g804 │ etcd/etcd-cilium             │ 2024-08-09 08:32:00 │ 365d         │ etcd-ca-int          │
│ node-master-hnf8g804 │ etcd/etcd-client             │ 2024-08-09 08:31:00 │ 365d         │ etcd-ca-int          │
│ node-master-hnf8g804 │ etcd/etcd-peer               │ 2024-08-09 08:31:01 │ 365d         │ etcd-ca-peer         │
│ node-master-hnf8g804 │ etcd/healthcheck-client      │ 2024-08-09 08:30:17 │ 365d         │ etcd-ca-int          │
│ node-master-hnf8g804 │ front-proxy-client           │ 2024-08-09 08:31:44 │ 365d         │ kubernetes-front-proxy-ca │
│ node-master-hnf8g804 │ kube-apiserver               │ 2024-08-09 08:31:43 │ 365d         │ kubernetes-ca        │
│ node-master-hnf8g804 │ kube-apiserver-etcd-client   │ 2024-08-09 08:31:44 │ 365d         │ etcd-ca-int          │
│ node-master-hnf8g804 │ kube-apiserver-kubelet-client │ 2024-08-09 08:31:43 │ 365d         │ kubernetes-ca        │
│ node-master-hnf8g804 │ kubelet                      │ 2024-08-09 08:31:45 │
```

```
365d          | kubernetes-kubelet-ca     |
| node-master-hnf8g804 | kubelet-client               | 2024-08-09 08:31:45 |
365d          | kubernetes-ca             |
| node-master-hnf8g804 | node-drainer                 | 2024-08-09 08:31:45 |
365d          | kubernetes-ca             |
| node-master-hnf8g804 | system-vault-secrets-webhook | 2024-08-09 08:30:22 |
365d          | kubernetes-ca             |
| node-master-hnf8g804 | users/admin                  | 2024-08-09 08:31:44 |
365d          | kubernetes-ca             |
| node-master-hnf8g804 | users/controller-manager     | 2024-08-09 08:31:44 |
365d          | kubernetes-ca             |
| node-master-hnf8g804 | users/scheduler              | 2024-08-09 08:31:45 |
365d          | kubernetes-ca             |
├──────────────────────┼──────────────────────────────┼─────────────────────┼──
──────────────┼──────────────────────────────┤
| node-worker-2ufpusql | etcd/etcd-cilium             | 2024-08-09 08:32:54 |
365d          | etcd-ca-int               |
| node-worker-2ufpusql | kubelet                      | 2024-08-09 08:30:58 |
365d          | kubernetes-kubelet-ca     |
| node-worker-2ufpusql | kubelet-client               | 2024-08-09 08:30:58 |
365d          | kubernetes-ca             |
| node-worker-2ufpusql | node-drainer                 | 2024-08-09 08:30:58 |
365d          | kubernetes-ca             |
├──────────────────────┼──────────────────────────────┼─────────────────────┼──
──────────────┼──────────────────────────────┤
| node-worker-zkot1gq2 | etcd/etcd-cilium             | 2024-08-09 08:32:42 |
365d          | etcd-ca-int               |
| node-worker-zkot1gq2 | kubelet                      | 2024-08-09 08:30:58 |
365d          | kubernetes-kubelet-ca     |
| node-worker-zkot1gq2 | kubelet-client               | 2024-08-09 08:30:58 |
365d          | kubernetes-ca             |
| node-worker-zkot1gq2 | node-drainer                 | 2024-08-09 08:30:58 |
365d          | kubernetes-ca             |
└──────────────────────┴──────────────────────────────┴─────────────────────┴──
──────────────┴──────────────────────────────┘
```

```
┏━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━┳━━━━
━━━━━━━━━━━━━┓
┃ Node                 ┃ Certificate authority        ┃ Expires             ┃
Residual time ┃
┣━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━╋━━━━
━━━━━━━━━━━━━┫
| node-master-hnf8g804 | etcd-ca-int                  | 2028-08-08 08:30:16 | 1825d
|
| node-master-hnf8g804 | etcd-ca-peer                 | 2028-08-08 08:30:17 | 1825d
|
| node-master-hnf8g804 | kubernetes-ingress           | 2028-08-08 08:30:24 | 1825d
|
| node-master-hnf8g804 | kubernetes-front-proxy-ca    | 2028-08-08 08:30:20 | 1825d
|
| node-master-hnf8g804 | kubernetes-kubelet-ca        | 2028-08-08 08:30:20 | 1825d
```

```
|
| node-master-hnf8g804 | kubernetes-signer-ca      | 2028-08-08 08:30:21 | 1825d
|
| node-master-hnf8g804 | kubernetes-ca             | 2028-08-08 08:30:20 | 1825d
|
| node-master-hnf8g804 | nova-platform.io          | 2033-08-07 08:29:58 | 3650d
|
└──────────────────────┴──────────────────────────┴─────────────────────┴──────
─────────┘
```

Для удобства срок действия сертификатов платформы предоставляется в виде двух таблиц:

- В первой таблице указаны серверные и клиентские сертификаты узлов платформы.
- Во второй таблице указаны сертификаты корневых центров.

2. Проверка сертификатов, выпущенных в Cert-Manager

Для проверки срока действия сертификатов, выпущенных с помощью Cert-Manager, администратор кластера может воспользоваться утилитой `kubectl` и следующей командой:

BASH

```
kubectl get certificate -A -o custom-
columns=NAME:.metadata.name,EXPIRES:.status.notAfter
```

Пример

BASH

```
$ kubectl get certificate -A -o custom-
columns=NAME:.metadata.name,EXPIRES:.status.notAfter
NAME                           EXPIRES
nova-vpa-admission-controller  2024-08-09T08:36:39Z
default-ingress-certificate    2024-08-09T08:36:37Z
nova-console-serving-cert      2024-08-09T08:36:37Z
monitoring-plugin-cert         2024-08-09T08:36:37Z
```

Также получить расширенную информацию о сертификатах возможно с помощью команды:

BASH

```
kubectl get certificate -A -o wide
```

Пример

BASH

```
$ kubectl get certificate -A -o wide
NAMESPACE             NAME                              READY   SECRET
```

```
ISSUER                                  STATUS
AGE
kube-system            nova-vpa-admission-controller   True    nova-vpa-
admission-controller-cert   nova-dynamic-internal-cluster-issuer   Certificate
is up to date and has not expired   21d
nova-cert-management   default-ingress-certificate     True    default-ingress-
certificate          nova-oauth-internal-cluster-issuer     Certificate is up to
date and has not expired   21d
nova-console           nova-console-serving-cert       True    nova-console-
serving-cert            nova-dynamic-internal-cluster-issuer   Certificate is up
to date and has not expired   21d
nova-monitoring        monitoring-plugin-cert          True    monitoring-
plugin-cert               nova-dynamic-internal-cluster-issuer   Certificate is
up to date and has not expired   21d
```

3. Следующие шаги

При необходимости вы можете обновить сертификаты платформы Nova Container Platform.

- Обновление сертификатов платформы

Управление цепочками сертификатов

В Nova Container Platform для управления цепочками TLS-сертификатов интегрировано решение Trust Manager.

Trust Manager - это оператор Kubernetes, предназначенный для централизованного управления и распространения цепочек доверенных сертификатов (*Trust Bundles*) в кластере Kubernetes. С его помощью обеспечивается унифицированное управление цепочками сертификатов, упрощается развертывание приложений, требующих доверия к определенным корневым удостоверяющим центрам (*CAs*), а также снижаются риски, связанные с использованием устаревших или недоверенных CA.

Цепочки Trust Manager в кластере Kubernetes - это ресурсы `Bundles` в API-группе `trust.cert-manager.io`.

1. Цепочка доверенных сертификатов по умолчанию

По умолчанию в кластере Kubernetes доступна цепочка `trusted-ca-bundle`, сертификаты которой сохранены в ConfigMap с таким же именем в родительском пространстве имен `nova-cert-management`.

Данная цепочка синхронизируется во все пространства имен, имеющие метку `nova-platform.io/trusted-ca-bundle: enabled`.

Источниками сертификатов для сборки всей цепочки служат ресурсы ConfigMap, имеющие метку `nova-platform.io/trusted-ca-bundle-inject: enabled` и ключ `ca.crt`, значение которого является сертификатом в формате PEM.

Несмотря на то, что в Trust Manager есть поддержка работы с секретами Kubernetes, в Nova Container Platform данная возможность отключена в целях безопасности и ограничения привилегий Trust Manager. Сертификаты CA не являются чувствительными данными и могут храниться в ConfigMap.

Пример цепочки по умолчанию

YAML

```yaml
apiVersion: trust.cert-manager.io/v1alpha1
kind: Bundle
metadata:
  name: trusted-ca-bundle
spec:
  sources:
    - useDefaultCAs: true
```

```
    - configMap:
        key: ca.crt
        selector:
          matchLabels:
            nova-platform.io/trusted-ca-bundle-inject: enabled
  target:
    configMap:
      key: ca-certificates.pem
    namespaceSelector:
      matchLabels:
        nova-platform.io/trusted-ca-bundle: enabled
```

Информация

Обратите внимание, что цепочка `trusted-ca-bundle` также включает публичные сертификаты (опция `useDefaultCAs: true`). Это означает, что используя цепочку по умолчанию, ваши сервисы будут доверять не только персональным сертификатам, но так же и всем публично доступным, включая сертификаты российский удостоверяющих центров.

2. Использование цепочек в пользовательских приложениях

Для использования цепочек сертификатов в собственных приложениях рекомендуется следующий порядок действий:

- Создать необходимые ресурсы ConfigMap с необходимыми сертификатами CA. Используйте один ресурс ConfigMap для одного сертификата CA. Установите метку на ресурс, по которой его можно будет найти оператору Trust Manager.
- Подготовьте необходимые пространства имен, в которые потребуется распространить цепочку сертификатов. Установите для этого соответствующую метку.
- Создать новый ресурс `Bundle`, где укажите параметры организации цепочки сертификатов. Вы можете указать несколько селекторов для поиска ресурсов ConfigMap с сертификатами.
- (Опционально) Включите в цепочку публичные сертификаты.

После того, как цепочка станет доступна в кластере Kubernetes, вы можете использовать ее в своих приложениях. Для этого вам необходимо будет смонтировать ConfigMap с цепочкой в Pod. В зависимости от ОС, которая лежит в основе контейнера, точка монтирования единого файла с сертификатами CA может отличаться:

- Для Debian-based ОС (Debian, Ubuntu), а также ОС Alpine, цепочку необходимо монтировать в файл `/etc/ssl/certs/ca-certificates.crt`.

- Для RHEL-based ОС (RHEL, CoreOS, Fedora, CentOS, AlmaLinux, Rocky Linux) цепочку необходимо монтировать в файл `/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem`.

Информация

При добавлении собственных сертификатов в цепочку по умолчанию `trusted-ca-bundle` некоторые из служебных сервисов Nova Container Platform будут автоматически перезапущены.

Организация инфраструктуры PKI в Nova Container Platform

В Kubernetes SSL/TLS сертификаты используются в различных сценариях для решения следующих задач:

- защита веб-трафика внутри кластера и за его пределами.
- дополнительная TLS-аутентификация между компонентами Kubernetes.

Следующие операции в среде Kubernetes требуют наличия инфраструктуры PKI:

- Клиентские сертификаты *Kubelet* необходимы для взаимодействия с *Kubernetes API*.
- Серверные сертификаты *Kubelet* необходимы *Kubernetes API* для взаимодействиями с *Kubelet* на узлах кластера.
- Серверные сертификаты необходимы для взаимодействия с *Kubernetes API*.
- Клиентские сертификаты администраторов кластера необходимы для взаимодействия с *Kubernetes API*.
- Клиентские сертификаты *Kubernetes API* необходимы для взаимодействиями с *Kubelet* на узлах кластера.
- Клиентские сертификаты *Kubernetes API* необходимы для взаимодействиями с хранилищем *Etcd*.
- Клиентские сертификаты *Controller Manager* необходимы для взаимодействиями с *Kubernetes API*.
- Клиентские сертификаты *Scheduler* необходимы для взаимодействиями с *Kubernetes API*.
- Клиентские сертификаты необходимы для компонента Kubernetes *Front Proxy*.
- Клиентские и серверные сертификаты ресурсов *Ingress* необходимы для взаимодействия пользователей платформы с опубликованными веб-ресурсами.

В хранилище *Etcd* также используется механизм взаимной TLS-аутентификации (mTLS) для клиентов и участников кластера.

В Nova Container Platform все необходимые для Kubernetes и компонентов платформы сертификаты создаются автоматически в StarVault и впоследствии могут быть автоматически или принудительно обновлены.

1. Архитектура PKI

Выпуск и управление требуемыми сертификатами осуществляется в *StarVault* с использованием движка *PKI Secrets Engine*. Данный движок позволяет не только динамически генерировать X.509 сертификаты, но также и организовывать центры сертификации (CA), обслуживать базы данных сертификатов, управлять процессами отзыва, а также применять политики сертификатов.

1.1. Центры сертификации

Центры сертификации в StarVault делятся на корневые и промежуточные:

- Корневые CA создаются по принципу Single root CA для каждого глобального блока платформы (Kubernetes, Etcd, Kubernetes Front Proxy и т.п.).
- Промежуточные CA используются в платформе более гранулярно, разделяя взаимодействие компонентов глобального блока платформы (Kubernetes API, Kubelet, Controller Manager и т.п.).

1.1.1. Корневые центры сертификации

В Nova Container Platform используется следующие корневые центры сертификации:

Имя	Default CN	TTL	Описание
`nova-etcd-pki-root`	`etcd-ca-root`	10 лет	Корневой CA для хранилища Etcd.
`nova-kubernetes-pki-root`	`kubernetes-ca-root`	10 лет	Корневой CA для инфраструктуры Kubernetes.

1.1.2. Промежуточные центры сертификации

В Nova Container Platform используется следующие промежуточные центры сертификации:

Имя	Default CN	Родительский CA	TTL	Описание
`nova-etcd-pki-int`	`etcd-ca-int`	`nova-etcd-pki-root`	5 лет	Промежуточный CA для хранилища Etcd. Применяется для выпуска сертификатов (Etcd Client).

Имя	Default CN	Родительский CA	TTL	Описание
`nova-etcd-pki-peer`	`etcd-ca-peer`	`nova-etcd-pki-root`	5 лет	Промежуточный CA для хранилища Etcd. Применяется для выпуска сертификатов (Etcd Peer).
`nova-kubernetes-pki-int`	`kubernetes-ca`	`nova-kubernetes-pki-root`	5 лет	Промежуточный CA для инфраструктуры Kubernetes. Применяется для выпуска сертификатов компонентов Kubernetes Control Plane.
`nova-kubernetes-pki-kubelet`	`kubernetes-kubelet-ca`	`nova-kubernetes-pki-root`	5 лет	Промежуточный CA для инфраструктуры Kubernetes. Применяется для выпуска сертификатов компонента Kubelet.
`nova-kubernetes-pki-ingress`	`kubernetes-ingress`	`nova-kubernetes-pki-root`	5 лет	Промежуточный CA для инфраструктуры Kubernetes. Применяется для выпуска сертификатов для ресурсов Ingress. Интегрирован с компонентом CertManager через ресурс *ClusterIssuer* `nova-oauth-internal-cluster-issuer`, а также может быть инициализирован с помощью пользовательского промежуточного сертификата.

Имя	Default CN	Родительский CA	TTL	Описание
`nova-kubernetes-pki-signer`	`kubernetes-signer-ca`	`nova-kubernetes-pki-root`	5 лет	Промежуточный CA для инфраструктуры Kubernetes. Применяется для выпуска сертификатов компонентом Controller Manager при использовании Certificates API. Интегрирован с компонентом CertManager через ресурс *ClusterIssuer* `nova-dynamic-internal-cluster-issuer` для автоматизированного выпуска сертификатов компонентов Kubernetes Admission Webhook.
`nova-kubernetes-pki-front-proxy`	`kubernetes-front-proxy-ca`	`nova-kubernetes-pki-root`	5 лет	Промежуточный CA для инфраструктуры Kubernetes. Используется в Kubernetes Front Proxy, применяется для выпуска сертификатов компонентов, расширяющих возможности Kubernetes API.

1.2. Политики выпуска сертификатов

Для обеспечения дополнительной безопасности процесса выпуска сертификатов в StarVault для каждого PKI автоматически настраиваются определенные политики выпуска (роли) сертификатов. Данные роли контролируют различные параметры выпускаемых сертификатов, например:

- Тип используемого криптографического алгоритма для ключей шифрования
- Разрешение на выпуск wildcard-сертификатов

- Правила проверки Common Name
- TTL
- Разрешенные домены
- Разрешение на добавление в сертификаты IP и DNS Sans и их перечень.
- Правила расширенного использования ключа (Extended Key Usage)

Каждая роль содержит только те параметры, которые требуются тому или иному компоненту согласно лучшим практикам Kubernetes. Таким образом, выпуск сертификата без строгого соответствия данным параметрам невозможен.

Некоторые параметры сертификатов в ролях являются динамическими и различаются в кластерах Nova Container Platform. При установке платформы `nova-ctl` генерирует данные параметры, используя данные конфигурационного манифеста, а затем выполняет инициализацию StarVault.

1.3. Расположение сертификатов

Корневые и промежуточные центры сертификации, создаваемые в StarVault, не являются экспортируемыми. Это означает, что сгенерированные приватные ключи CA сохраняются в StarVault в зашифрованном виде и впоследствии не могут быть получены.

На узлах кластера Kubernetes находятся выпущенные сертификаты и приватные ключи для компонентов Kubernetes в следующих директориях и файлах:

- `/opt/nova/conf.d/pki/`
- `/opt/nova/conf.d/pki/etcd/`