Red Hat OpenStack Services on OpenShift 18.0

配置安全服务

在 OpenShift 中为 Red Hat OpenStack Services 配置安全功能

Last Updated: 2024-10-07

Red Hat OpenStack Services on OpenShift 18.0 配置安全服务

在 OpenShift 中为 Red Hat OpenStack Services 配置安全功能

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

根据您的环境要求，在 OpenShift 上为 Red Hat OpenStack Services 自定义安全功能。

目录

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。与我们分享您的成功秘诀。

在 JIRA 中提供文档反馈

使用 Create Issue 表单在 OpenShift (RHOSO)或更早版本的 Red Hat OpenStack Platform (RHOSP)上提供有关 Red Hat OpenStack Services 文档的反馈。当您为 RHOSO 或 RHOSP 文档创建问题时，这个问题将在 RHOSO Jira 项目中记录，您可以在其中跟踪您的反馈的进度。

要完成 Create Issue 表单，请确保您已登录到 JIRA。如果您没有红帽 JIRA 帐户，您可以在 https://issues.redhat.com 创建一个帐户。

1. 点击以下链接打开 **Create Issue** 页面： Create Issue
2. 完成 **Summary** 和 **Description** 字段。在 **Description** 字段中，包含文档 URL、章节号以及问题的详细描述。不要修改表单中的任何其他字段。
3. 点 **Create**。

第 1 章 在 OPENSHIFT 中为 RED HAT OPENSTACK SERVICES 添加自定义 TLS 证书

当您在 OpenShift (RHOSO)上部署 Red Hat OpenStack Services 时，默认启用 TLS。TLS 由 *cert-manager* 处理，它同时应用入口（公共）加密，以及重新加密每个 pod。目前，不支持在 RHOSO 上禁用 TLS。

1.1. RED HAT OPENSTACK SERVICES ON OPENSHIFT 中的 TLS

当您在 OpenShift 上部署 Red Hat OpenStack Services (RHOSO)时，大多数 API 连接都受到 TLS 的保护。

注意

TLS 目前不适用于内部 Alert Manager Web UI 服务端点。

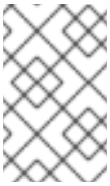
您可能需要使用自己的内部证书颁发机构保护公共 API。要替换自动生成的证书，您必须创建一个包含额外 ca certs 的 secret，包括在所需的信任链中的所有证书。

您可以将您自己的内部证书颁发机构(CA)中的可信证书应用到 RHOSO 上的公共接口。公共接口是入口流量满足服务的路由。不要试图管理内部(pod 级别)接口上的加密。

如果您决定应用您自己的内部证书颁发机构(CA)的可信证书，则需要以下信息。

DNS 名称

对于您要应用自己的自定义证书的每个服务，您将需要其 DNS 主机名来生成证书。您可以使用以下命令获取公共主机名列表： **oc get -n openstack routes**



注意

要将单个证书用于两个或多个服务，请在 DNS 名称字段中使用通配符，或者在 subject alt name 字段中列出多个 DNS 名称。**如果不使用通配符，则必须在路由主机名更改时更新证书。**

Duration

要在 OpenShift 中更新服务的证书，必须重启该服务。证书持续时间是服务在不重启的情况下可以保持最长的时间，受您的内部安全策略。

usages

您必须在证书的 **使用列表中包含 - 密钥加密** 加密、**数字签名 和服务器 auth**。

更新 TLS 以使用自定义证书，需要对 control plane 和数据平面进行编辑。

以下是在没有注解和更改时使用的默认 TLS 设置：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: myctlplane
spec:
  tls:
    default:
```

```
  ingress:
    ca:
      duration: 87600h
    cert:
      duration: 43800h
    enabled: true
  podLevel:
    enabled: true
    internal:
      ca:
        duration: 87600h
      cert:
        duration: 43800h
    libvirt:
      ca:
        duration: 87600h
      cert:
        duration: 43800h
    ovn:
      ca:
        duration: 87600h
      cert:
        duration: 43800h
```

- 要为每个公共服务创建自定义 TLS 证书，请参阅使用公共服务的自定义证书更新 control plane。
- 要创建应用到公共服务的单个自定义 TLS 证书，请参阅为公共服务使用单个自定义证书更新 control plane。

1.2. 使用公共服务的自定义证书更新 CONTROL PLANE

您可能需要使用自己的内部证书颁发机构(CA)保护公共 API。要将自动生成的路由证书替换为来自您的 CA 的通用证书，您必须创建一个包含额外 CA 证书的 secret，以及信任链中的所有证书。

先决条件

- 您有每个公共服务的列表，用于应用您的自定义服务证书。您可以使用 **oc route list -n openstack** 命令获取此列表。对于您必须创建的证书数量、这些证书的 DNS 名称，以及在 **openstack_control_plane.yaml** 自定义资源(CR)中查找要编辑的相关服务，请使用此信息。
- 您有一个用于公共服务的服务证书

流程

1. 创建名为 **cacerts.yaml** 的清单文件，其中包含所有 CA 证书。如果需要，在信任链中包括所有证书：

```
apiVersion: v1
kind: Secret
metadata:
  name: cacerts
  namespace: openstack
type: Opaque
```

```
data:
  myBundleExample: <cat mybundle.pem | base64 -w0> (1)
  CACertExample: <cat cacert.pem | base64 -w0>     (2)
```

1. 运行这个命令，将 **mybundle.pem** 替换为证书或证书捆绑包的名称。结果被粘贴为 **myBundleExample** 字段的值。
2. 运行这个命令，将 **cacert.pem** 替换为您的 CA 证书的名称。

2. 从清单文件创建 secret：

```
oc apply -f cacerts.yaml
```

3. 为每个 secret 创建一个名为 **api_certificate_<service>_secret.yaml 的清单文件** ：

```
apiVersion: v1
kind: Secret
metadata:
  name: api_certificate_<service>_secret (1)
  namespace: openstack
type: kubernetes.io/tls
data:
  tls.crt: <cat tlscrt.pem | base64 -w0> (2)
  tls.key: <cat tlskey.pem | base64 -w0> (3)
  ca.crt: <cat cacrt.pem | base64 -w0>   (4)
```

1. 将 **<service** > 替换为此 secret 所在的服务的名称。
2. 运行这个命令，将 **tlscrt.pem** 替换为签名证书的名称。
3. 运行这个命令，将 **tlskey.pem** 替换为私钥的名称。
4. 运行这个命令，将 **cacrt.pem** 替换为 CA 证书的名称。

4. 创建 secret

```
oc apply -f api_certificate_<service>_secret.yaml
```

5. 编辑 **openstack_control_plane.yaml** 自定义资源，并将捆绑包添加为 **caBundleSecretName** 的参数：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: myctlplane
spec:
  tls:
    podLevel:
      enabled: true
    caBundleSecretName: cacerts
```

6. 将 secret 服务证书应用到 apiOverride 字段中的每个公共服务。例如，为 Identity 服务(keystone)输入以下内容：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: myctlplane
  namespace: openstack
spec:
  ...
  keystone:
    apiOverride:
      tls:
        secretName: api_certificate_keystone_secret
```

Compute 服务(nova)和 NoVNCProxy 的编辑如下所示：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: myctlplane
  namespace: openstack
spec:
...
  nova:
    apiOverride:
      tls:
        secretName: api_certificate_nova_secret
      route: {}
    cellOverride:
      cell1:
        NoVNCProxy:
          tls:
            secretName: api_certificate_novavncproxy_secret
```

7. 应用 control plane 更改

```
oc apply -f openstack_control_plane.yaml
```

1.3. 使用单个自定义证书为公共服务更新 CONTROL PLANE

您可能需要使用自己的内部证书颁发机构(CA)保护公共 API。要将自动生成的路由证书替换为来自您的 CA 的通用证书，您必须创建一个包含 CA 证书的 secret，以及信任链中的所有证书。

先决条件

- 您有每个公共服务的列表，用于应用您的自定义服务证书。您可以使用 **oc route list -n openstack** 命令获取此列表。将此信息用于证书的 DNS 名称，并在 **openstack_control_plane.yaml** 自定义资源(CR)中查找要编辑的相关服务。

流程

1. 在 **alt_names** 部分中创建一个包含每个服务的主机名的签名证书：

```
[alt_names]
DNS.1 = barbican-public-openstack.apps.ocp.openstack.lab
DNS.2 = cinder-public-openstack.apps.ocp.openstack.lab
DNS.3 = glance-default-public-openstack.apps.ocp.openstack.lab
DNS.4 = horizon-openstack.apps.ocp.openstack.lab
DNS.5 = keystone-public-openstack.apps.ocp.openstack.lab
DNS.6 = manila-public-openstack.apps.ocp.openstack.lab
DNS.7 = neutron-public-openstack.apps.ocp.openstack.lab
DNS.8 = nova-novncproxy-cell1-public-openstack.apps.ocp.openstack.lab
DNS.9 = nova-public-openstack.apps.ocp.openstack.lab
DNS.10 = placement-public-openstack.apps.ocp.openstack.lab
```

2. 创建名为 **cacerts.yaml** 的清单文件，其中包含所有 CA 证书。如果需要，在信任链中包括所有证书：

```
apiVersion: v1
kind: Secret
metadata:
  name: cacerts
  namespace: openstack
type: Opaque
data:
  myBundleExample: <cat mybundle.pem | base64 -w0> (1)
  CACertExample: <cat cacert.pem | base64 -w0>  (2)
```

(1) 运行这个命令，将 **mybundle.pem** 替换为证书或证书捆绑包的名称。结果被粘贴为 **myBundleExample** 字段的值。

(2) 运行这个命令，将 **cacert.pem** 替换为您的 CA 证书的名称。

3. 从清单文件创建 secret：

```
oc apply -f cacerts.yaml
```

4. 为名为 **certificate-secret.yaml** 的 secret 创建清单文件：

```
apiVersion: v1
kind: Secret
metadata:
  name: certificate-secret
  namespace: openstack
type: kubernetes.io/tls
data:
  tls.crt: <cat tlscrt.pem | base64 -w0> (1)
  tls.key: <cat tlskey.pem | base64 -w0> (2)
  ca.crt: <cat cacrt.pem | base64 -w0>  (3)
```

(1) 运行这个命令，将 **tlscrt.pem** 替换为签名证书的名称。

(2) 运行这个命令，将 **tlskey.pem** 替换为私钥的名称。

(3) 运行这个命令，将 **cacrt.pem** 替换为 CA 证书的名称。

5. 创建 secret

```
oc apply -f certificate-secret.yaml
```

6. 编辑 **openstack_control_plane.yaml** 自定义资源，并将捆绑包添加为 **caBundleSecretName** 的参数：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: myctlplane
spec:
  tls:
    podLevel:
      enabled: true
    caBundleSecretName: cacerts
```

7. 将 secret 服务证书应用到 **apiOverride** 字段中的每个公共服务。例如，为 Identity 服务 (keystone)输入以下内容：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: myctlplane
  namespace: openstack
spec:
  ...
  keystone:
    apiOverride:
      tls:
        secretName: certificate-secret
```

Compute 服务(nova)和 **NoVNCProxy** 的编辑如下所示：

```
apiVersion: core.openstack.org/v1beta1
kind: OpenStackControlPlane
metadata:
  name: myctlplane
  namespace: openstack
spec:
...
  nova:
    apiOverride:
      tls:
        secretName: certificate-secret
      route: {}
    cellOverride:
      cell1:
        NoVNCProxy:
          tls:
            secretName: certificate-secret
```

8. 应用 control plane 更改

```
oc apply -f openstack_control_plane.yaml
```