Red Hat OpenStack Services on OpenShift 18.0

Configuring security services

Configuring the security features for Red Hat OpenStack Services on OpenShift

Last Updated: 2024-08-19

Red Hat OpenStack Services on OpenShift 18.0 Configuring security services

Configuring the security features for Red Hat OpenStack Services on OpenShift

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

Customize security features for Red Hat OpenStack Services on OpenShift based on the requirements of your environment.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Tell us how we can make it better.

Providing documentation feedback in Jira

Use the Create Issue form to provide feedback on the documentation for Red Hat OpenStack Services on OpenShift (RHOSO) or earlier releases of Red Hat OpenStack Platform (RHOSP). When you create an issue for RHOSO or RHOSP documents, the issue is recorded in the RHOSO Jira project, where you can track the progress of your feedback.

To complete the Create Issue form, ensure that you are logged in to Jira. If you do not have a Red Hat Jira account, you can create an account at https://issues.redhat.com.

1. Click the following link to open a **Create Issue** page: Create Issue
2. Complete the **Summary** and **Description** fields. In the **Description** field, include the documentation URL, chapter or section number, and a detailed description of the issue. Do not modify any other fields in the form.
3. Click **Create**.

CHAPTER 1. ADDING CUSTOM TLS CERTIFICATES FOR RED HAT OPENSTACK SERVICES ON OPENSHIFT

When you deploy Red Hat OpenStack Services on OpenShift (RHOSO), TLS-e (TLS everywhere) is enabled by default. TLS is handled by *cert-manager*, which applies both ingress (public) encryption, as well as reencryption to each pod. Currently, disabling TLS on RHOSO is not supported.

1.1. TLS IN RED HAT OPENSTACK SERVICES ON OPENSHIFT

When you deploy Red Hat OpenStack Services on OpenShift (RHOSO), most API connections are protected by TLS.

NOTE

TLS is not currently available for the internal Alert Manager Web UI service endpoint.

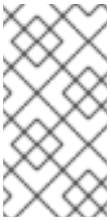
You might be required to protect public APIs using your own internal certificate authority. In order to replace the automatically generated certificates you must create a secret that contains your additional ca certs, including all certificates in needed chains of trust.

You can apply trusted certificates from your own internal certificate authority (CA) to public interfaces on RHOSO. The public interface is where ingress traffic meets the service's route. Do not attempt to manage encryption on internal (pod level) interfaces.

If you decide to apply trusted certificates from your own internal certificate authority (CA), you will need the following information.

DNS names

For each service you apply your own custom certificate to, you will need its DNS hostname for the process of generating the certificate. You can get a list of public hostnames using the following command: **oc get -n openstack routes**



NOTE

To use a single certificate for two or more services, use a wildcard in the DNS name field, or list multiple DNS names in the subject alt names field. **If you do not use a wildcard, then you must update the certificate in the event of a route hostname change.**

Duration

To update a service's certificate in OpenShift, the service must be restarted. The duration for the certificate is the longest amount of time a service can stay live without being restarted, subject to your internal security policies.

Usages

You must include - **key encipherment**, **digital signature**, and **server auth** within the list of usages in your certificate.

Updating TLS to use custom certificates requires edits to both the control plane and the data plane.

The following is the default TLS settings that are used if not annotated and changed:

```
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
```

- To create a custom TLS certificate for each public service see Updating the control plane with custom certificates for public services.
- To create a single custom TLS certificate to apply to the public services, see proc_updating-the-control-plane-with-single-certificate.adoc.

1.2. UPDATING THE CONTROL PLANE WITH CUSTOM CERTIFICATES FOR PUBLIC SERVICES

You might be required to protect public APIs by using your own internal certificate authority (CA). To replace the automatically generated route certificates with common signed certificates from your CA, you must create a secret that contains your additional CA certificate, and all certificates in the chain of trust.

Prerequisites

- You have a list of each of the public services for which to apply your custom service certificates. You can get this list using the **oc route list -n openstack** command. Use this information for the number of certificates you must create, the DNS names for those certificates, as well as finding the relevant services to edit in the **openstack_control_plane.yaml** custom resource (CR).
- You have a service certificate for the public services

Procedure

1. Create a manifest file called **cacerts.yaml** that includes all CA certificates. Include all certificates in chains of trust if applicable:

```
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. Run this command, replacing **mybundle.pem** with the name of your certificate or certificate bundle. The results are pasted as the value of the **myBundleExample** field.
2. Run this command, replacing **cacert.pem** with the name of your CA certificate.

2. Create the secret from the manifest file:

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

3. Create a manifest file for each secret named **api_certificate_<service>_secret.yaml**:

```
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. Replace **<service>** with the name of the service that this secret is for.
2. Run this command, replacing **tlscrt.pem** with the name of your signed certificate.
3. Run this command, replacing **tlskey.pem** with the name of your private key.
4. Run this command, replacing **cacrt.pem** with the name of your CA certificate.

4. Create the secret

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

5. Edit the **openstack_control_plane.yaml** custom resource and add your bundle as the parameter for **caBundleSecretName**:

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

6. Apply the secret service certificates to each of the public services under the apiOverride field. For example enter the following for the Identity service (keystone):

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

The edits for the Compute service (nova) and NoVNCProxy appear as the following:

```
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. Apply the control plane changes

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

1.3. UPDATING THE CONTROL PLANE WITH A SINGLE CUSTOM CERTIFICATES FOR PUBLIC SERVICES

You might be required to protect public APIs by using your own internal certificate authority (CA). To replace the automatically generated route certificates with a common signed certificate from your CA, you must create a secret that contains your CA certificate, and all certificates in the chain of trust.

Prerequisites

- You have a list of each of the public services for which to apply your custom service certificate. You can get this list by using the **oc route list -n openstack** command. Use this information for the DNS names for the certificate, as well as for finding the relevant services to edit in the **openstack_control_plane.yaml** custom resource (CR).

Procedure

1. Create a signed certificate that includes the hostname for every service in the **alt_names** section:

```
[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. Create a manifest file called **cacerts.yaml** that includes all CA certificates. Include all certificates in chains of trust if applicable:

```
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. Run this command, replacing **mybundle.pem** with the name of your certificate or certificate bundle. The results are pasted as the value of the **myBundleExample** field.

2. Run this command, replacing **cacert.pem** with the name of your CA certificate.

3. Create the secret from the manifest file:

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

4. Create a manifest file for a secret named **certificate-secret.yaml**:

```
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. Run this command, replacing **tlscrt.pem** with the name of your signed certificate.
2. Run this command, replacing **tlskey.pem** with the name of your private key.
3. Run this command, replacing **cacrt.pem** with the name of your CA certificate.

5. Create the secret

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

6. Edit the **openstack_control_plane.yaml** custom resource and add your bundle as the parameter for **caBundleSecretName**:

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

7. Apply the secret service certificates to each of the public services under the **apiOverride** field. For example, enter the following for the Identity service (keystone):

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

The edits for the Compute service (nova) and **NoVNCProxy** appear as the following:

```
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. Apply the control plane changes

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