IBM Cloud Private 3.1.2

Lab Exercise # Security 1

Enable TLS with custom CA Issuer and certificate at Ingress

Duration: 45 mins

Objective

The objective of this lab is to know how to enable TLS communication in the applications. Also get to know different ways of using certificates in IBM Cloud Private environment and manage the Issuers.

Pre-requisites

We are going to use a helloworld application for this lab.

Deploy a helloworld app using the YAML file provided below.

Note: Please change the service name, which is highlighted in the section below, before making the deployment.

kubectl apply -f <yaml file> --namespace <your-namespace>

```
apiVersion: v1
kind: Service
metadata:
  name: helloworld<your-user-id>
  labels:
    app: helloworld
spec:
  ports:
  - port: 5000
    name: http
  selector:
    app: helloworld
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: helloworld-v1
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: helloworld
        version: v1
    spec:
      containers:
      - name: helloworld
        image: ilon1.icp:8500/helloworld-v1
        resources:
          requests:
            cpu: "100m"
        imagePullPolicy: IfNotPresent #Always
        ports:
        - containerPort: 5000
```

Instructions

1. Check the Issuer in the IBM Cloud Private environment.

```
$ kubectl get clusterissuer
NAME          AGE
icp-ca-issuer   132d
```

Get the issuer specific to the namespace, if any available

```
$ kubectl get issuer
NAME                  AGE
hello-deployment-tls   1h
hello-self-tls         1h
```

In the next step let's create a custom "Issuer" specific to namespace and use that in "Ingress" to access the helloworld application.

2. Create a self-signed Issuer. Use the following .yaml file to define a self-signed Issuer.

   Note: Replce **<your-namespace>** and **<user-id>** with the namespace given to you, in the yaml given below.

kubectl apply -f <yaml file> --namespace <your-namespace>

```
apiVersion: certmanager.k8s.io/v1alpha1
kind: Issuer
metadata:
 name: hello-<user-id>-tls
 namespace: <your-namespace>
spec:
 selfSigned: {}
```

3. After you create the self-signed Issuer, create a CA certificate that references the self-signed Issuer and specifies the isCA field.

   Note: Replce **<user-id>** with the user-id given to you, in the yaml given below.

kubectl apply -f <yaml file> --namespace <your-namespace>

```
apiVersion: certmanager.k8s.io/v1alpha1
kind: Certificate
metadata:
 name: hello-<user-id>-cert-tls
spec:
 # name of the tls secret to store
 # the generated certificate/key pair
 secretName: hello-<user-id>-tls-ca-key-pair
 isCA: true
 issuerRef:
   # issuer created in step 1
   name: hello-<user-id>-tls
   kind: Issuer
 commonName: "foo<user-id>.bar"
 dnsNames:
 # one or more fully-qualified domain name
 # can be defined here
 - foo<user-id>.bar
```

4. Edit the following sample of an Issuer that references the previous secret. Edit the `<name>` and `<namespace>` from the *metadata* section of the `.yaml` file. Be sure that `secretName` from the *spec* section matches the `secretName` from the previous step:

   Note: Replce **<your-namespace>** and **<user-id>** with the namespace given to you, in the yaml given below.

kubectl apply -f <yaml file> --namespace <your-namespace>

```
apiVersion: certmanager.k8s.io/v1alpha1
kind: Issuer
metadata:
 name: self-ca-issuer-<user-id>
 namespace: blueworld
spec:
 ca:
   secretName: hello-<user-id>-tls-ca-key-pair
```

Now a custom issuer called 'self-ca-issuer-<user-id>' is created, you can create custom CA issuer with third party or enterprise provider.

5. Define the certificate using the custom CA issuer created in Step 3.
   `helloworld.x.x.x.x.nip.io is the CN for the certificate. X.X.X.X is the IP address of proxy`

   Note: Replce **<your-namespace>** and **<user-id>** with the namespace given to you, in the yaml given below.

kubectl apply -f <yaml file> --namespace <your-namespace>

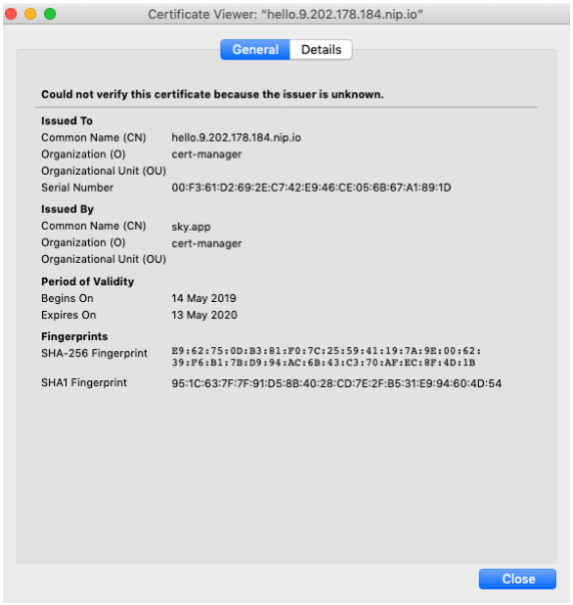
```
apiVersion: certmanager.k8s.io/v1alpha1
kind: Certificate
metadata:
 name: helloworld-tls-cert-<user-id>
 namespace: <your-namespace>
spec:
 # name of the tls secret to store
 # the generated certificate/key pair
 secretName: helloworld-tls-certs-<user-id>
 isCA: true
 issuerRef:
   # issuer created in step 1
   name: self-ca-issuer-<user-id>
   kind: Issuer
 commonName: helloworld<user-id>.172.16.70.58.nip.io
 dnsNames:
 # one or more fully-qualified domain name
 # can be defined here
 - helloworld<user-id>.172.16.70.58.nip.io
```

6. Add the Secret to the Kubernetes Ingress. The following step defines a TLS-enabled Kubernetes Ingress that is integrated with cert-manager. Here, **helloworld-tls-certs** matches the secretName that you previously defined and host matches the DNS name that you previously defined in the certificate.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: helloworld-<user-id>-ingress-tls
 annotations:
   kubernetes.io/ingress.class: "nginx"
   ingress.kubernetes.io/rewrite-target: "/"
spec:
 tls:
 # k8s ingress defines different tls certificates
 # for each nginx server blocks.
 # k8s ingress default cert is used if
 # no host-specific secret specified
 - hosts:
 # this is the fully-qualified domain name
 # of the first server block
   - hello.9.202.178.184.nip.io
   # certificate hello-k8s-ingress-tls-1
   # is only used by foo1.bar1
   secretName: helloworld-tls-certs-<user-id>
 rules:
   # each server block redirects request
   # to its own backend service
 - host: helloworld<user-id>.172.16.70.58.nip.io
   http:
     paths:
     - backend:
         serviceName: helloworld
         servicePort: 5000
```

7. Access the application using https://helloworld.x.x.x.x.nip.io/hello

If you are using self-signed key, probably a warning message on the certificate would appear, try to view the detail of the certificate



Summary

You have gone through the steps of deploying Ingress with self-signed certificate generated through custom CA Issuer.

Try the following

- Enabling SSL with third-party certificates for IBM Cloud Private and deploy the Liberty server

https://www.ibm.com/support/knowledgecenter/SSD28V_9.0.0/com.ibm.websphere.wlp.core.doc/ae/twlp_icp_auto_ssl3.html

- Try to deploy liberty application with default ICP issuer

https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.1.2/manage_applications/create_issuer.html