



Exercise 15.1: Working with TLS

Overview

We have learned that the flow of access to a cluster begins with TLS connectivity, then authentication followed by authorization, finally an admission control plug-in allows advanced features prior to the request being fulfilled. The use of `Initializers` allows the flexibility of a shell-script to dynamically modify the request. As security is an important, ongoing concern, there may be multiple configurations used depending on the needs of the cluster.

Every process making API requests to the cluster must authenticate or be treated as an anonymous user.

While one can have multiple cluster root Certificate Authorities (CA) by default each cluster uses their own, intended for intra-cluster communication. The CA certificate bundle is distributed to each node and as a secret to default service accounts. The **kubelet** is a local agent which ensures local containers are running and healthy.

1. View the **kubelet** on both the cp and secondary nodes. The **kube-apiserver** also shows security information such as certificates and authorization mode. As **kubelet** is a **systemd** service we will start looking at that output.

```
student@cp:~$ systemctl status kubelet.service
```

```
1 kubelet.service - kubelet: The Kubernetes Node Agent
2   Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: en
3   Drop-In: /etc/systemd/system/kubelet.service.d
4           |__10-kubeadm.conf
5  <output_omitted>
```

2. Look at the status output. Follow the CGroup and kubelet information, which is a long line where configuration settings are drawn from, to find where the configuration file can be found.

```
1 CGroup: /system.slice/kubelet.service
2   |--19523 /usr/bin/kubelet .... --config=/var/lib/kubelet/config.yaml ..
```

3. Take a look at the settings in the `/var/lib/kubelet/config.yaml` file. Among other information we can see the `/etc/kubernetes/pki/` directory is used for accessing the **kube-apiserver**. Near the end of the output it also sets the directory to find other pod spec files.

```
student@cp:~$ sudo less /var/lib/kubelet/config.yaml
```

YAML

config.yaml

```
1 <output_omitted>
2 rotateCertificates: true
3 runtimeRequestTimeout: 0s
4 shutdownGracePeriod: 0s
5 shutdownGracePeriodCriticalPods: 0s
6 staticPodPath: /etc/kubernetes/manifests
7 streamingConnectionIdleTimeout: 0s
8 syncFrequency: 0s
9 volumeStatsAggPeriod: 0s
```

4. Other agents on the cp node interact with the **kube-apiserver**. View the configuration files where these settings are made. This was set in the previous YAML file. Look at one of the files for cert information.

```
student@cp:~$ sudo ls /etc/kubernetes/manifests/
```

```
1 etcd.yaml kube-controller-manager.yaml
2 kube-apiserver.yaml kube-scheduler.yaml
```

```
student@cp:~$ sudo less /etc/kubernetes/manifests/kube-controller-manager.yaml
```

```
1 <output_omitted>
```

5. The use of tokens has become central to authorizing component communication. The tokens are kept as **secrets**. Take a look at the current secrets in the kube-system namespace.

```
student@cp:~$ kubectl -n kube-system get secrets
```

```
1 NAME                                     TYPE
2 DATA      AGE
3 attachdetach-controller-token-xqr8n    kubernetes.io/service-account-token
4 3      5d
5 bootstrap-signer-token-xbp6s           kubernetes.io/service-account-token
6 3      5d
7 bootstrap-token-i3r13t                 bootstrap.kubernetes.io/token
8 7      5d
9 <output_omitted>
```

6. Take a closer look at one of the secrets and the token within. The certificate-controller-token could be one to look at. The use of the Tab key can help with long names. Long lines have been truncated in the output below.

```
student@cp:~$ kubectl -n kube-system get secrets certificate<Tab> -o yaml
```

YAML

```
1 apiVersion: v1
2 data:
3   ca.crt: LS0tLS1CRUdJT...
4   namespace: a3ViZS1zeXNOZW0=
5   token: ZXlKaGJHY2lPaUpTVXpJM...
6 kind: Secret
7 metadata:
8   annotations:
9     kubernetes.io/service-account.name: certificate-controller
10    kubernetes.io/service-account.uid: 7dfa2aa0-9376-11e8-8cfb
11    -42010a800002
12   creationTimestamp: 2018-07-29T21:29:36Z
13   name: certificate-controller-token-wnrwh
14   namespace: kube-system
15   resourceVersion: "196"
16   selfLink: /api/v1/namespaces/kube-system/secrets/certificate-
17   controller-token-wnrwh
18   uid: 7dfbb237-9376-11e8-8cfb-42010a800002
19 type: kubernetes.io/service-account-token
```

7. The **kubectl config** command can also be used to view and update parameters. When making updates this could avoid a typo removing access to the cluster. View the current configuration settings. The keys and certs are redacted from the output automatically.

```
student@cp:~$ kubectl config view
```

```
1 apiVersion: v1
2 clusters:
3 - cluster:
4   certificate-authority-data: REDACTED
5 <output_omitted>
```

8. View the options, such as setting a password for the admin instead of a key. Read through the examples and options.

```
student@cp:~$ kubectl config set-credentials -h
```

```
1 Sets a user entry in kubeconfig
2 <output_omitted>
```

9. Make a copy of your access configuration file. Later steps will update this file and we can view the differences.

```
student@cp:~$ cp $HOME/.kube/config $HOME/cluster-api-config
```

10. Explore working with cluster and security configurations both using **kubectl** and **kubeadm**. Among other values, find the name of your cluster. You will need to become root to work with **kubeadm**.

```
student@cp:~$ kubectl config <Tab><Tab>
```

```
1 current-context  get-contexts      set-context      view
2 delete-cluster  rename-context    set-credentials
3 delete-context  set               unset
4 get-clusters    set-cluster       use-context
```

```
student@cp:~$ sudo kubeadm token -h
```

```
1 <output_omitted>
```

```
student@cp:~$ sudo kubeadm config -h
```

```
1 <output_omitted>
```

11. Review the cluster default configuration settings. There may be some interesting tidbits to the security and infrastructure of the cluster.

```
student@cp:~$ sudo kubeadm config print init-defaults
```

```
1 apiVersion: kubeadm.k8s.io/v1beta2
2 bootstrapTokens:
3 - groups:
4   - system:bootstrappers:kubeadm:default-node-token
5   token: abcdef.0123456789abcdef
6   ttl: 24h0m0s
7   usages:
8 <output_omitted>
```