**CKA LAB PART 5 - SECURITY**

**Lab 1 - RBAC within a namespace**

Implement the following:

• Create the namespace “rbac-test”

• Create the service account “rbac-test-sa” for the “rbac-test” namespace

• Create a role “rbac-test-role” that grants the following pod level resources:

o Get

o Watch

o List

• Bind the “rbac-test-sa” service account to the “rbac-test-role” role

• Test RBAC is working by trying to do something the service account is not authorised to do

**Lab 2 - RBAC within a cluster**

Implement the following:

• Create the user “cluster-user-secretadmin” authenticating with a password

• Create a role “cluster-role-secretadmin” that grants the following cluster level secret resources:

o Get

o Watch

o List

• Bind “cluster-user-secretadmin” user to the “cluster-role-secretadmin”

Note:

# Switch to the admin context

kubectl config use-context kubernetes-admin@kubernetes

# Apply the ClusterRole

kubectl apply -f cluster-role-secretadmin.yaml

# Apply the ClusterRoleBinding

kubectl apply -f cluster-rolebinding-secretadmin.yaml

# Verify the creation

kubectl get clusterroles | grep cluster-role-secretadmin

kubectl get clusterrolebindings | grep cluster-rolebinding-secretadmin

# Switch back to the original context

kubectl config use-context cluster-user-secretadmin-context

**Lab 3 - Network security policy**

• Create a nginx pod that listens on port 80, note the IP assigned to it.

• Create two pods that can use “curl” named busybox1 and busybox2. Note the IP addresses assigned to them. Label them with tier:jumppod

• Take a interactive shell to busybox1 and run:

o Curl [IP Address of nginx pod]. You should get an HTML response

• Create a NetworkPolicy rule that blocks all ingress traffic to the nginx pod

• Rerun the curl command from busybox1, it should fail.

• Create a NetworkPolicy that blocks all ingress traffic to the nginx pod with the exception of all pods labelled with tier:jumppod

**Lab 4 - Enable Pod Security Policy**

Configure the admission controller in your cluster to use PodSecurityPolicy

**Lab 5 - Create policies**

Create two pod security policies

• One named “Privileged” with no restrictions

• One named “Restricted” with the following restrictions

o Cannot run privileged containers

o Can only be exposed on port 433

**Lab 6 - Security Context**

Create a pod that defines subsequent containers to run as a user id of 600

**Lab 7 - Secure persistent key value store**

• Generate a key that will be used to encrypt information located in etcd and create the respective configuration file

• Modify the API server to leverage the encryption configuration leveraging the key generated in

step 1

• Create a secret called “testsecret” via any applicable means. Verfy the contents are encrypted.