Free Questions for CKS by certsinside

Shared by Payne on 06-06-2022

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

You can switch the cluster/configuration context using the following command:

[desk@cli] $kubectl config use-context dev

Context:

A CIS Benchmark tool was run against the kubeadm created cluster and found multiple issues that must be addressed.

Task:

Fix all issues via configuration and restart the affected components to ensure the new settings take effect.

Fix all of the following violations that were found against the API server:

1.2.7authorization-modeargument is not set toAlwaysAllow FAIL

1.2.8authorization-modeargument includesNode FAIL

1.2.7authorization-modeargument includesRBAC FAIL

Fix all of the following violations that were found against the Kubelet:

4.2.1 Ensure that theanonymous-auth argumentis set to false FAIL

4.2.2authorization-modeargument is not set to AlwaysAllow FAIL (UseWebhookautumn/authz where possible)

Fix all of the following violations that were found against etcd:

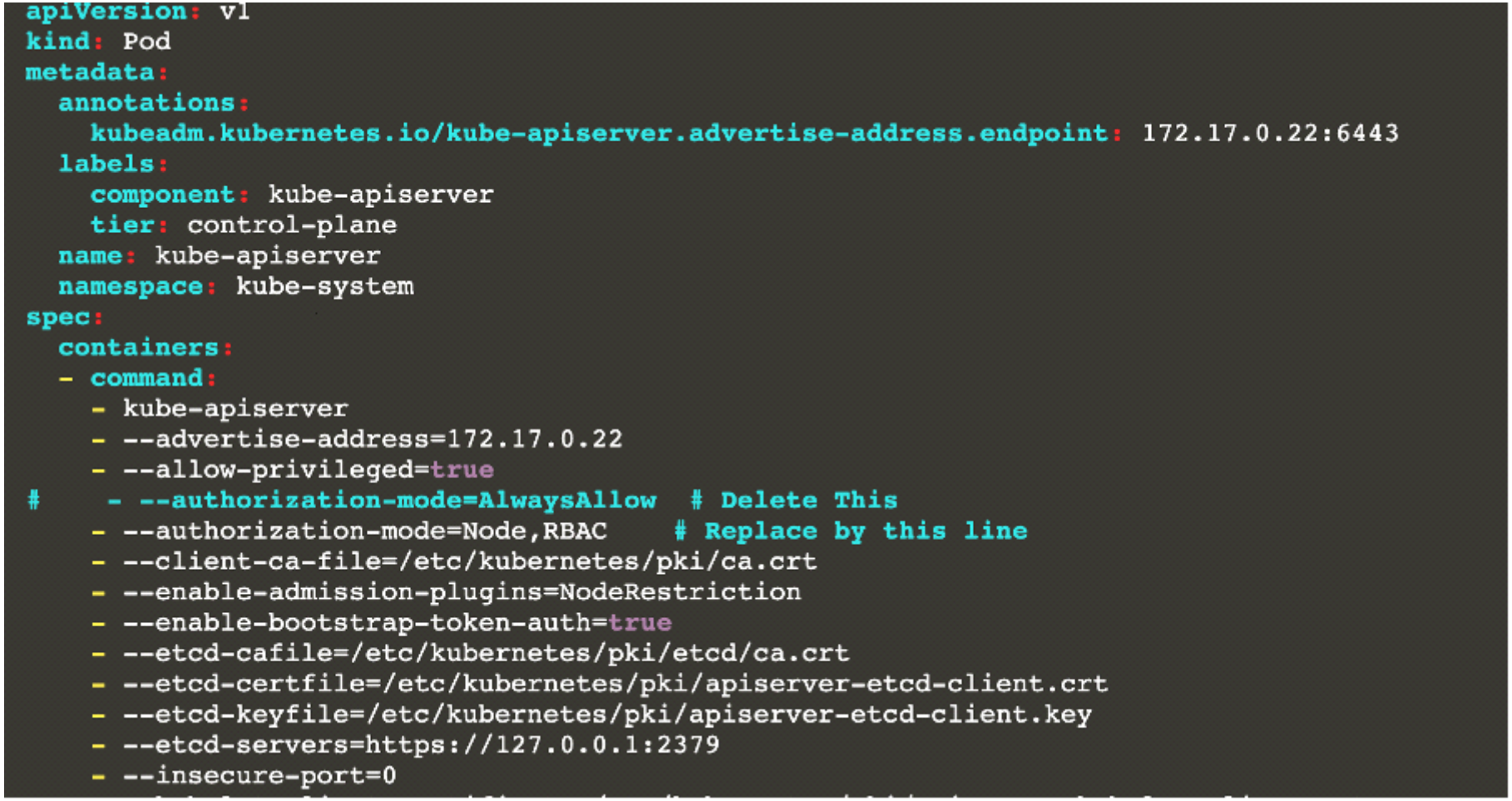
2.2 Ensure that theclient-cert-authargument is set to true

Options:

A- Explanation:
worker1 $ vim /var/lib/kubelet/config.yaml
anonymous:
enabled: true #Delete this
enabled: false #Replace by this
authorization:
mode: AlwaysAllow #Delete this
mode: Webhook #Replace by this
worker1 $ systemctl restart kubelet. # To reload kubelet config
ssh to master1
master1 $ vim /etc/kubernetes/manifests/kube-apiserver.yaml
- -- authorization-mode=Node,RBAC
master1 $ vim /etc/kubernetes/manifests/etcd.yaml
- --client-cert-auth=true
Explanation
ssh to worker1
worker1 $ vim /var/lib/kubelet/config.yaml

```
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
anonymous:
enabled: true #Delete this
enabled: false #Replace by this
webhook:
cacheTTL: 0s
enabled: true
x509:
clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
mode: AlwaysAllow #Delete this
mode: Webhook #Replace by this
webhook:
cacheAuthorizedTTL: 0s
cacheUnauthorizedTTL: 0s
cgroupDriver: systemd
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
```

imageMinimumGCAge: 0s
kind: KubeletConfiguration
logging: {}
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
resolvConf: /run/systemd/resolve/resolv.conf
rotateCertificates: true
runtimeRequestTimeout: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s
worker1 $ systemctl restart kubelet. # To reload kubelet config
ssh to master1
master1 $ vim /etc/kubernetes/manifests/kube-apiserver.yaml



```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 172.17.0.22:6443
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=172.17.0.22
    - --allow-privileged=true
#     - --authorization-mode=AlwaysAllow  # Delete This
    - --authorization-mode=Node,RBAC    # Replace by this line
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
```

master1 $ vim /etc/kubernetes/manifests/etcd.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/etcd.advertise-client-urls: https://172.17.0.29:2379
  creationTimestamp: null
  labels:
    component: etcd
    tier: control-plane
  name: etcd
  namespace: kube-system
spec:
  containers:
  - command:
    - etcd
    - --advertise-client-urls=https://172.17.0.29:2379
    - --cert-file=/etc/kubernetes/pki/etcd/server.crt
    - --client-cert-auth=true  #Add this line
    - --data-dir=/var/lib/etcd
    - --initial-advertise-peer-urls=https://172.17.0.29:2380
    - --initial-cluster=controlplane=https://172.17.0.29:2380
    - --key-file=/etc/kubernetes/pki/etcd/server.key
    - --listen-client-urls=https://127.0.0.1:2379,https://172.17.0.29:2379
    - --listen-metrics-urls=http://127.0.0.1:2381
    - --listen-peer-urls=https://172.17.0.29:2380
    - --name=controlplane
    - --peer-cert-file=/etc/kubernetes/pki/etcd/peer.crt
    - --peer-client-cert-auth=true
    - --peer-key-file=/etc/kubernetes/pki/etcd/peer.key
    - --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
    - --snapshot-count=10000
    - --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
    image: k8s.gcr.io/etcd:3.4.9-1
    imagePullPolicy: IfNotPresent
```

Answer:

A

Question 2

Question Type: MultipleChoice

You can switch the cluster/configuration context using the following command:

[desk@cli] $kubectl config use-context prod-account

Context:

A Role bound to a Pod's ServiceAccount grants overly permissive permissions. Complete the following tasks to reduce the set of permissions.

Task:

Given an existing Pod namedweb-podrunning in the namespacedatabase.

1. Edit the existing Role bound to the Pod's ServiceAccounttest-sato only allow performing get operations, only on resources of type Pods.

2. Create a new Role namedtest-role-2in the namespacedatabase, which only allows performingupdateoperations, only on resources of typestatuefulsets.

3. Create a new RoleBinding namedtest-role-2-bindbinding the newly created Role to the Pod's ServiceAccount.

Note: Don't delete the existing RoleBinding.

Options:

A- Explanation:
$k edit role test-role -n database
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
creationTimestamp: '2021-06-04T11:12:23Z'
name: test-role
namespace: database
resourceVersion: '1139'
selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/database/roles/test-role
uid: 49949265-6e01-499c-94ac-5011d6f6a353
rules:
- apiGroups:
- ''
resources:
- pods

verbs:
- * # Delete
- get # Fixed
$k create role test-role-2 -n database --resource statefulset --verb update
$k create rolebinding test-role-2-bind -n database --role test-role-2 --serviceaccount=database:test-sa
Explanation
[desk@cli]$k get pods -n database
NAME READY STATUS RESTARTS AGE LABELS
web-pod 1/1 Running 0 34s run=web-pod
[desk@cli]$k get roles -n database
test-role
[desk@cli]$k edit role test-role -n database
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
creationTimestamp: '2021-06-13T11:12:23Z'
name: test-role
namespace: database
resourceVersion: '1139'
selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/database/roles/test-role
uid: 49949265-6e01-499c-94ac-5011d6f6a353
rules:
- apiGroups:
- ''
resources:
- pods

verbs:
- '*' # Delete this
- get # Replace by this
[desk@cli]$k create role test-role-2 -n database --resource statefulset --verb update
role.rbac.authorization.k8s.io/test-role-2 created

[desk@cli]$k create rolebinding test-role-2-bind -n database --role test-role-2 --serviceaccount=database:test-sa
rolebinding.rbac.authorization.k8s.io/test-role-2-bind created

Reference:https://kubernetes.io/docs/reference/access-authn-authz/rbac/
role.rbac.authorization.k8s.io/test-role-2 created
[desk@cli]$k create rolebinding test-role-2-bind -n database --role test-role-2 --serviceaccount=database:test-sa
rolebinding.rbac.authorization.k8s.io/test-role-2-bind created
[desk@cli]$k create role test-role-2 -n database --resource statefulset --verb update
role.rbac.authorization.k8s.io/test-role-2 created

[desk@cli]$k create rolebinding test-role-2-bind -n database --role test-role-2 --serviceaccount=database:test-sa
rolebinding.rbac.authorization.k8s.io/test-role-2-bind created

Reference:https://kubernetes.io/docs/reference/access-authn-authz/rbac/

Answer:

A

Question 3

Question Type: MultipleChoice

Context:

Cluster:gvisor

Master node:master1

Worker node:worker1

You can switch the cluster/configuration context using the following command:

[desk@cli] $kubectl config use-context gvisor

Context:This cluster has been prepared to support runtime handler, runsc as well as traditional one.

Task:

Create a RuntimeClass namednot-trustedusing the prepared runtime handler namesrunsc.

Update all Pods in the namespace server to run onnewruntime.

Options:

A- Explanation:

Find all the pods/deployment and edit runtimeClassName parameter to not-trusted under spec

```
[desk@cli] $k edit deploy nginx
spec:
runtimeClassName: not-trusted. # Add this
```

Explanation

```
[desk@cli] $vim runtime.yaml
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
name: not-trusted
handler: runsc
[desk@cli] $k apply -f runtime.yaml
[desk@cli] $k get pods
NAME READY STATUS RESTARTS AGE
nginx-6798fc88e8-chp6r 1/1 Running 0 11m
nginx-6798fc88e8-fs53n 1/1 Running 0 11m
nginx-6798fc88e8-ndved 1/1 Running 0 11m
[desk@cli] $k get deploy
NAME READY UP-TO-DATE AVAILABLE AGE
nginx 3/3 11 3 5m
[desk@cli] $k edit deploy nginx
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  strategy: {}
  template:
    metadata:
      labels:
        app: nginx
    spec:
      runtimeClassName: not-trusted     # Add this
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
```

Answer:

A

Question 4

Question Type: **MultipleChoice**

You can switch the cluster/configuration context using the following command:

[desk@cli] $kubectl config use-context dev

A default-deny NetworkPolicy avoid to accidentally expose a Pod in a namespace that doesn't have any other NetworkPolicy defined.

Task: Create a new default-deny NetworkPolicy nameddeny-networkin the namespacetestfor all traffic of type Ingress + Egress

The new NetworkPolicy must deny all Ingress + Egress traffic in the namespacetest.

Apply the newly createddefault-denyNetworkPolicy to all Pods running in namespacetest.

You can find a skeleton manifests file at /home/cert_masters/network-policy.yaml

Options:

A- Explanation:

master1 $k get pods -n test --show-labels

NAME READY STATUS RESTARTS AGE LABELS

test-pod 1/1 Running 0 34s role=test,run=test-pod

testing 1/1 Running 0 17d run=testing

$vim netpol.yaml

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: deny-network

namespace: test

spec:

podSelector: {}

policyTypes:

- Ingress

- Egress

master1 $k apply -f netpol.yaml

Explanation

controlplane $ k get pods -n test --show-labels

NAME READY STATUS RESTARTS AGE LABELS

test-pod 1/1 Running 0 34s role=test,run=test-pod

testing 1/1 Running 0 17d run=testing

master1 $ vim netpol1.yaml

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: deny-network
namespace: test
spec:
podSelector: {}
policyTypes:
- Ingress
- Egress
master1 $ k apply -f netpol1.yaml

Reference:
https://kubernetes.io/docs/concepts/services-networking/network-policies/

Answer:

A

Explanation:

master1 $ k apply -f netpol1.yaml Reference: https://kubernetes.io/docs/concepts/services-networking/network-policies/

Explanation

controlplane $ k get pods -n test --show-labels

NAME READY STATUS RESTARTS AGE LABELS

test-pod 1/1 Running 0 34s role=test,run=test-pod

testing 1/1 Running 0 17d run=testing

master1 $ vim netpol1.yaml

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: deny-network

namespace: test

spec:

podSelector: {}

policyTypes:

- Ingress

- Egress

master1 $ k apply -f netpol1.yaml Reference: https://kubernetes.io/docs/concepts/services-networking/network-policies/

master1 $ k apply -f netpol1.yaml Reference: https://kubernetes.io/docs/concepts/services-networking/network-policies/

Question 5

Question Type: **MultipleChoice**

Cluster:scanner

Master node:controlplane

Worker node:worker1

You can switch the cluster/configuration context using the following command:

[desk@cli] $kubectl config use-context scanner

Given:

You may use Trivy's documentation.

Task:

Use the Trivy open-source container scanner to detect images with severe vulnerabilities used by Pods in the namespacenato.

Look for images withHighorCriticalseverity vulnerabilities and delete the Pods that use those images.

Trivy is pre-installed on the cluster's master node. Use cluster's master node to use Trivy.

Options:

A- Explanation:

[controlplane@cli] $k get pods -n nato -o yaml | grep 'image: '

[controlplane@cli] $trivy image <image-name>

[controlplane@cli] $k delete pod <vulnerable-pod> -n nato

[desk@cli] $ssh controlnode

[controlplane@cli] $k get pods -n nato

NAME READY STATUS RESTARTS AGE

alohmora 1/1 Running 0 3m7s

c3d3 1/1 Running 0 2m54s

neon-pod 1/1 Running 0 2m11s

thor 1/1 Running 0 58s

[controlplane@cli] $k get pods -n nato -o yaml | grep 'image: '

```
controlplane $ k get pods -n nato -o yaml | grep "image:"
                f:image: {}
    - image: alpine:3.12
      image: alpine:3.12
                f:image: {}
    - image: alpine:3.12
      image: alpine:3.12
                f:image: {}
    - image: alpine:3.7
      image: alpine:3.7
                f:image: {}
    - image: nginx
      image: nginx:latest
```

[controlplane@cli] $k delete pod thor -n nato
[controlplane@cli] $k delete pod neon-pod -n nato

Reference:https://github.com/aquasecurity/trivy
[controlplane@cli] $k delete pod neon-pod -n nato

Reference:https://github.com/aquasecurity/trivy

Answer:

A

Question 6

Question Type: **MultipleChoice**

Cluster: dev

Master node:master1

Worker node:worker1

You can switch the cluster/configuration context using the following command:

[desk@cli] $kubectl config use-context dev

Task:

Retrieve the content of the existing secret namedadamin thesafenamespace.

Store the username field in a file names/home/cert-masters/username.txt, and the password field in a file named/home/cert-masters/password.txt.

1. You must create both files; they don't exist yet.

2. Do not use/modify the created files in the following steps, create new temporary files if needed.

Create a new secret namesnewsecretin thesafenamespace, with the following content:

Username:dbadmin

Password:moresecurepas

Finally, create a new Pod that has access to the secretnewsecretvia a volume:

Namespace: safe

Pod name: mysecret-pod

Container name: db-container

Image: redis

Volume name: secret-vol

Mount path: /etc/mysecret

Options:

A- Explanation:

1. Get the secret, decrypt it & save in files

k get secret adam -n safe -o yaml

2. Create new secret using --from-literal

[desk@cli] $k create secret generic newsecret -n safe --from-literal=username=dbadmin --from-literal=password=moresecurepass

3. Mount it as volume of db-container of mysecret-pod

Explanation

```
controlplane $ k get secret adam -n safe -o yaml
apiVersion: v1
data:
  password: c2VjcmV0cGFzcw==
  username: Y2VydC1tYXN0ZXJz
kind: Secret
metadata:
  creationTimestamp: "2021-06-13T11:56:32Z"
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:data:
        .: {}
        f:password: {}
        f:username: {}
      f:type: {}
    manager: kubectl-create
    operation: Update
    time: "2021-06-13T11:56:32Z"
  name: adam
  namespace: safe
  resourceVersion: "6405"
```

```
controlplane $ echo "c2VjcmV0cGFzcw==" | base64 -d | tee -a /home/cert-masters/password.txt
secretpasscontrolplane $
```

[desk@cli] $k create secret generic newsecret -n safe --from-literal=username=dbadmin --from-literal=password=moresecurepass
secret/newsecret created
[desk@cli] $vim /home/certs_masters/secret-pod.yaml
apiVersion: v1
kind: Pod
metadata:
name: mysecret-pod
namespace: safe
labels:
run: mysecret-pod
spec:
containers:
- name: db-container
image: redis
volumeMounts:
- name: secret-vol
mountPath: /etc/mysecret
readOnly: true
volumes:
- name: secret-vol
secret:
secretName: newsecret
[desk@cli] $k apply -f /home/certs_masters/secret-pod.yaml

pod/mysecret-pod created

[desk@cli] $k exec -it mysecret-pod -n safe -- cat /etc/mysecret/username

dbadmin

```
controlplane $ k exec -it mysecret-pod -n safe -- cat /etc/mysecret/username
dbadmincontrolplane $
```

[desk@cli] $k exec -it mysecret-pod -n safe -- cat /etc/mysecret/password

moresecurepas

```
controlplane $ k exec -it mysecret-pod -n safe -- cat /etc/mysecret/password
moresecurepasscontrolplane $
```

Answer:

A

Question 7

Question Type: **MultipleChoice**

Context:

Cluster:prod

Master node:master1

Worker node:worker1

You can switch the cluster/configuration context using the following command:

[desk@cli] $kubectl config use-context prod

Task:

Analyse and edit the given Dockerfile (based on theubuntu:18:04image)

/home/cert_masters/Dockerfilefixing two instructions present in the file being prominent security/best-practice issues.

Analyse and edit the given manifest file

/home/cert_masters/mydeployment.yamlfixing two fields present in the file being prominent security/best-practice issues.

Note:Don't add or remove configuration settings; only modify the existing configuration settings, so that two configuration settings each are no longer security/best-practice concerns.

Should you need an unprivileged user for any of the tasks, use usernobodywith user id65535

Options:

A- Explanation:

1. For Dockerfile:Fix the image version & user name in Dockerfile
2. For mydeployment.yaml : Fix security contexts

Explanation

[desk@cli] $vim /home/cert_masters/Dockerfile

FROM ubuntu:latest # Remove this

FROM ubuntu:18.04 # Add this

USER root # Remove this

USER nobody # Add this

RUN apt get install -y lsof=4.72 wget=1.17.1 nginx=4.2

ENV ENVIRONMENT=testing

USER root # Remove this

USER nobody # Add this

CMD ['nginx -d']

```
FROM ubuntu:latest  # Remove this
FROM ubuntu:18.04   # Add this
USER root           # Remove this
USER nobody         # Add this
RUN apt get install -y lsof=4.72 wget=1.17.1 nginx=4.2
ENV  ENVIRONMENT=testing
USER root           # Remove this
USER nobody         # Add this
CMD ["nginx -d"]
```

[desk@cli] $vim/home/cert_masters/mydeployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:
creationTimestamp: null
labels:
app: kafka
name: kafka
spec:
replicas: 1
selector:
matchLabels:
app: kafka
strategy: {}
template:
metadata:
creationTimestamp: null
labels:
app: kafka
spec:
containers:
- image: bitnami/kafka
name: kafka
volumeMounts:
- name: kafka-vol
mountPath: /var/lib/kafka
securityContext:
{'capabilities':{'add':['NET_ADMIN'],'drop':['all']},'privileged': True,'readOnlyRootFilesystem': False, 'runAsUser': 65535} # Delete This
{'capabilities':{'add':['NET_ADMIN'],'drop':['all']},'privileged': False,'readOnlyRootFilesystem': True, 'runAsUser': 65535} # Add This

resources: {}
volumes:
- name: kafka-vol
emptyDir: {}
status: {}
Pictorial View:
[desk@cli] $vim/home/cert_masters/mydeployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: kafka
  name: kafka
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kafka
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: kafka
    spec:
      containers:
      - image: bitnami/kafka
        name: kafka
        volumeMounts:
          - name: kafka-vol
            mountPath: /var/lib/kafka
        securityContext:
          {"capabilities":{"add":["NET_ADMIN"],"drop":["all"]},"privileged": True,"readOnlyRootFilesystem": False, "runAsUser": 65535} # Delete This
          {"capabilities":{"add":["NET_ADMIN"],"drop":["all"]},"privileged": False,"readOnlyRootFilesystem": True, "runAsUser": 65535} # Add This
        resources: {}
      volumes:
        - name: kafka-vol
          emptyDir: {}
status: {}
```

Answer:

A

Question 8

Question Type: **MultipleChoice**

You can switch the cluster/configuration context using the following command:

[desk@cli] $kubectl config use-context stage

Context:

A PodSecurityPolicy shall prevent the creation of privileged Pods in a specific namespace.

Task:

1. Create a new PodSecurityPolcy named deny-policy, which prevents the creation of privileged Pods.

2. Create a new ClusterRole name deny-access-role, which uses the newly created PodSecurityPolicy deny-policy.

3. Create a new ServiceAccount named psd-denial-sa in the existing namespace development.

Finally, create a new ClusterRoleBindind named restrict-access-bind, which binds the newly created ClusterRole deny-access-role to the newly created ServiceAccount psp-denial-sa

Options:

A- Explanation:

Create psp to disallow privileged container

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

name: deny-access-role

rules:

- apiGroups: ['policy']

resources: ['podsecuritypolicies']

verbs: ['use']

resourceNames:

- "deny-policy"

k create sa psp-denial-sa -n development

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: restrict-access-bing

roleRef:

kind: ClusterRole

name: deny-access-role

apiGroup: rbac.authorization.k8s.io

subjects:

- kind: ServiceAccount

name: psp-denial-sa

namespace: development
Explanation
master1 $ vim psp.yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
name: deny-policy
spec:
privileged: false # Don't allow privileged pods!
seLinux:
rule: RunAsAny
supplementalGroups:
rule: RunAsAny
runAsUser:
rule: RunAsAny
fsGroup:
rule: RunAsAny
volumes:
- '*'
master1 $ vim cr1.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: deny-access-role
rules:
- apiGroups: ['policy']

resources: ['podsecuritypolicies']
verbs: ['use']
resourceNames:
- "deny-policy"
master1 $k create sa psp-denial-sa -n development
master1 $ vim cb1.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
name: restrict-access-bing
roleRef:
kind: ClusterRole
name: deny-access-role
apiGroup: rbac.authorization.k8s.io
subjects:
Authorize specific service accounts:
- kind: ServiceAccount
name: psp-denial-sa
namespace: development
master1 $k apply -f psp.yaml
master1 $k apply -f cr1.yaml
master1 $k apply -f cb1.yaml

Reference:https://kubernetes.io/docs/concepts/policy/pod-security-policy/
master1 $k apply -f cr1.yaml
master1 $k apply -f cb1.yaml

master1 $k apply -f psp.yaml
master1 $k apply -f cr1.yaml
master1 $k apply -f cb1.yaml

Reference:https://kubernetes.io/docs/concepts/policy/pod-security-policy/

Answer:

A

Question 9

Question Type: **MultipleChoice**

You must complete this task on the following cluster/nodes: Cluster:immutable-cluster

Master node:master1

Worker node:worker1

You can switch the cluster/configuration context using the following command:

[desk@cli] $kubectl config use-context immutable-cluster

Context: It is best practice to design containers to be stateless and immutable.

Task:

Inspect Pods running in namespaceprodand delete any Pod that is either not stateless or not immutable.

Use the following strict interpretation of stateless and immutable:

1. Pods being able to store data inside containers must be treated as not stateless.

Note:You don't have to worry whether data is actually stored inside containers or not already.

2. Pods being configured to beprivilegedin any way must be treated as potentially not stateless or not immutable.

Options:

A- Explanation:
k get pods -n prod
k get pod -n prod -o yaml | grep -E 'privileged|ReadOnlyRootFileSystem'
Delete the pods which do have any of these 2 properties
privileged:trueorReadOnlyRootFileSystem: false
[desk@cli]$k get pods -n prod
NAME READY STATUS RESTARTS AGE
cms 1/1 Running 0 68m
db 1/1 Running 0 4m
nginx 1/1 Running 0 23m
[desk@cli]$k get pod nginx -n prod -o yaml | grep -E 'privileged|RootFileSystem'
{'apiVersion':'v1','kind':'Pod','metadata':{'annotations':{},'creationTimestamp':null,'labels':{'run':'nginx'},'name':'nginx','namespace':'prod'},'spec':{'contai

f:privileged: {}

privileged:true

```
controlplane $ k get pod nginx -n prod -o yaml | grep -E 'privileged|RootFileSystem'
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"creationTimestamp":null,"labels":{"run":"nginx"},"name":"nginx","namespace":"prod"},"spec":{"containers":[{"image":"ngi
nx","name":"nginx","resources":{},"securityContext":{"privileged":true}}],"dnsPolicy":"ClusterFirst","restartPolicy":"Always"},"status":{}}
            f:privileged: {}
      privileged: true
```

[desk@cli]$k delete pod nginx -n prod

[desk@cli]$k get pod db -n prod -o yaml | grep -E 'privileged|RootFilesystem'

```
controlplane $ k get pod db -n prod -o yaml  | grep -E 'privileged|RootFileSystem'
controlplane $
```

[desk@cli]$k delete pod cms -n prod

Reference:https://kubernetes.io/docs/concepts/policy/pod-security-policy/

https://cloud.google.com/architecture/best-practices-for-operating-containers

Answer:

A

Explanation:

[desk@cli]$k delete pod cms -n prod Reference:https://kubernetes.io/docs/concepts/policy/pod-security-policy/

https://cloud.google.com/architecture/best-practices-for-operating-containers

Question 10

Question Type: MultipleChoice

Cluster:admission-cluster

Master node:master

Worker node:worker1

You can switch the cluster/configuration context using the following command:

[desk@cli] $kubectl config use-context admission-cluster

Context:

A container image scanner is set up on the cluster, but it's not yet fully integrated into the cluster's configuration. When complete, the container image scanner shall scan for and reject the use of vulnerable images.

Task:

You have to complete the entire task on the cluster's master node, where all services and files have been prepared and placed.

Given an incomplete configuration in directory /etc/Kubernetes/config and a functional container image scanner with HTTPS endpoint https://imagescanner.local:8181/image_policy:

1. Enable the necessary plugins to create an image policy

2. Validate the control configuration and change it to an implicit deny

3. Edit the configuration to point to the provided HTTPS endpoint correctly

Finally, test if the configuration is working by trying to deploy the vulnerable resource /home/cert_masters/test-pod.yml

Note: You can find the container image scanner's log file at /var/log/policy/scanner.log

Options:

A- Explanation:
[master@cli] $cd /etc/Kubernetes/config
1. Edit kubeconfig to explicity deny
[master@cli] $vim kubeconfig.json
'defaultAllow': false # Change to false
2. fix server parameter by taking its value from ~/.kube/config
[master@cli] $cat /etc/kubernetes/config/kubeconfig.yaml | grep server
server:
3. EnableImagePolicyWebhook
[master@cli] $vim /etc/kubernetes/manifests/kube-apiserver.yaml
- --enable-admission-plugins=NodeRestriction,ImagePolicyWebhook # Add this
- --admission-control-config-file=/etc/kubernetes/config/kubeconfig.json # Add this
Explanation
[desk@cli] $ssh master
[master@cli] $cd /etc/Kubernetes/config
[master@cli] $vim kubeconfig.json

{
'imagePolicy': {
'kubeConfigFile': '/etc/kubernetes/config/kubeconfig.yaml',
'allowTTL': 50,
'denyTTL': 50,
'retryBackoff': 500,
'defaultAllow': true # Delete this
'defaultAllow': false # Add this
}
}

```
{
  "imagePolicy": {
    "kubeConfigFile": "/etc/kubernetes/config/kubeconfig.yaml",
    "allowTTL": 50,
    "denyTTL": 50,
    "retryBackoff": 500,
    "defaultAllow": true    # Delete this
    "defaultAllow": false   # Add this
  }
}
```

Note:We can see a missing value here, so how from where i can get this value
[master@cli] $cat ~/.kube/config | grep server
or
[master@cli] $cat /etc/kubernetes/manifests/kube-apiserver.yaml

```
controlplane $ cat ~/.kube/config | grep server
    server: https://172.17.0.36:6443
```

[master@cli] $vim /etc/kubernetes/config/kubeconfig.yaml

```
apiVersion: v1
kind: Config
clusters:
  - cluster:
      certificate-authority: /etc/kubernetes/config/ca.pem
      server: https://172.17.0.36:6443    #Add this
    name: kubernetes
  - cluster:
contexts:
- context:
    cluster: kubernetes
    user: kube-admin
  name: webhook
current-context: webhook
users:
  - name: kube-admin
    user:
      client-certificate: /etc/kubernetes/config/cert.pem
      client-key: /etc/kubernetes/config/key.pem
```

[master@cli] $vim /etc/kubernetes/manifests/kube-apiserver.yaml

- --enable-admission-plugins=NodeRestriction # Delete This
- --enable-admission-plugins=NodeRestriction,ImagePolicyWebhook # Add this
- --admission-control-config-file=/etc/kubernetes/config/kubeconfig.json # Add this

Reference:https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/
- --enable-admission-plugins=NodeRestriction # Delete This
- --enable-admission-plugins=NodeRestriction,ImagePolicyWebhook # Add this
- --admission-control-config-file=/etc/kubernetes/config/kubeconfig.json # Add this
[master@cli] $vim /etc/kubernetes/manifests/kube-apiserver.yaml
- --enable-admission-plugins=NodeRestriction # Delete This
- --enable-admission-plugins=NodeRestriction,ImagePolicyWebhook # Add this
- --admission-control-config-file=/etc/kubernetes/config/kubeconfig.json # Add this

Reference:https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/

Answer:

A

Question 11

Question Type: **MultipleChoice**

Cluster:qa-cluster

Master node:masterWorker node:worker1

You can switch the cluster/configuration context using the following command:

[desk@cli] $kubectl config use-context qa-cluster

Task:

Create a NetworkPolicy namedrestricted-policyto restrict access to Podproductrunning in namespacedev.

Only allow the following Pods to connect to Pod products-service:

1. Pods in the namespaceqa

2. Pods with labelenvironment: stage, in any namespace

Options:

A- Explanation:
$k get ns qa --show-labels
NAME STATUS AGE LABELS
qa Active 47m env=stage
$k get pods -n dev --show-labels
NAME READY STATUS RESTARTS AGE LABELS
product 1/1 Running 0 3s env=dev-team

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: restricted-policy
namespace: dev
spec:
podSelector:
matchLabels:
env: dev-team
policyTypes:
- Ingress
ingress:
- from:
- namespaceSelector:
matchLabels:
env: stage
- podSelector:
matchLabels:
env: stage
[desk@cli] $k get ns qa --show-labels
NAME STATUS AGE LABELS
qa Active 47m env=stage
[desk@cli] $k get pods -n dev --show-labels
NAME READY STATUS RESTARTS AGE LABELS
product 1/1 Running 0 3s env=dev-team
[desk@cli] $vim netpol2.yaml
```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: restricted-policy
namespace: dev
spec:
podSelector:
matchLabels:
env: dev-team
policyTypes:
- Ingress
ingress:
- from:
- namespaceSelector:
matchLabels:
env: stage
- podSelector:
matchLabels:
env: stage
[desk@cli] $k apply -f netpol2.yaml

Reference:https://kubernetes.io/docs/concepts/services-networking/network-policies/
[desk@cli] $k apply -f netpol2.yaml

Reference:https://kubernetes.io/docs/concepts/services-networking/network-policies/

Answer:

A

Question 12

Question Type: MultipleChoice

Using the runtime detection tool Falco, Analyse the container behavior for at least 20 seconds, using filters that detect newly spawning and executing processes in a single container of Nginx.

Options:

A- store the incident file art /opt/falco-incident.txt, containing the detected incidents. one per line, in the format [timestamp],[uid],[processName]

Answer:

A