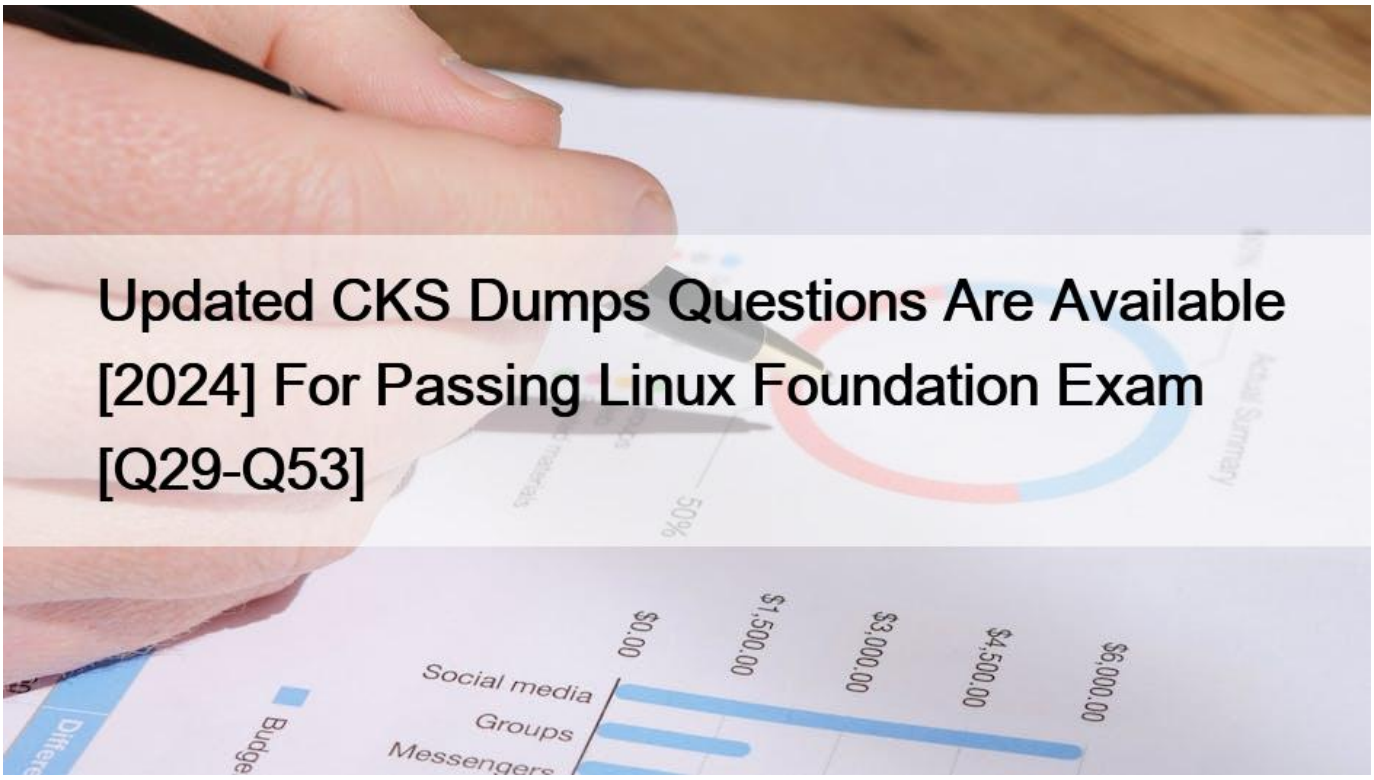
Updated CKS Dumps Questions Are Available [2024 For Passing Linux Foundation Exam [Q29-Q53



Updated CKS Dumps Questions Are Available [2024] For Passing Linux Foundation Exam
Free UPDATED Linux Foundation CKS Certification Exam Dumps is Online

NO.29 SIMULATION

A container image scanner is set up on the cluster.

Given an incomplete configuration in the directory

/etc/Kubernetes/confcontrol and a functional container image scanner with HTTPS endpoint https://acme.local.8081/image_policy

1. Enable the admission plugin.

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

Finally, test the configuration by deploying the pod having the image tag as the latest.
* Send us the Feedback on it.

NO.30 A container image scanner is set up on the cluster.

Given an incomplete configuration in the directory

/etc/kubernetes/confcontrol and a functional container image scanner with HTTPS endpoint
https://test-server.local.8081/image_policy

1. Enable the admission plugin.

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

Finally, test the configuration by deploying the pod having the image tag as latest.
ssh-add ~/.ssh/tempprivate

eval “$(ssh-agent -s)”

cd contrib/terraform/aws

vi terraform.tfvars

terraform init

terraform apply -var-file=credentials.tfvars

ansible-playbook -i ./inventory/hosts ./cluster.yml -e ansible_ssh_user=core -e bootstrap_os=coreos -b –become-user=root –flush-cache -e ansible_user=core

NO.31 Service is running on port 389 inside the system, find the process-id of the process, and stores the names of all the open-files inside the /candidate/KH77539/files.txt, and also delete the binary.
* Send us your Feedback on this.

NO.32 Fix all issues via configuration and restart the affected components to ensure the new setting takes effect.

Fix all of the following violations that were found against the API server:- a. Ensure the –authorization-mode argument includes RBAC b. Ensure the –authorization-mode argument includes Node c. Ensure that the –profiling argument is set to false Fix all of the following violations that were found against the Kubelet:- a. Ensure the –anonymous-auth argument is set to false.

b. Ensure that the –authorization-mode argument is set to Webhook.

Fix all of the following violations that were found against the ETCD:-

a. Ensure that the –auto-tls argument is not set to true

Hint: Take the use of Tool Kube-Bench
API server:

Ensure the –authorization-mode argument includes RBAC

Turn on Role Based Access Control. Role Based Access Control (RBAC) allows fine-grained control over the operations that different entities can perform on different objects in the cluster. It is recommended to use the RBAC authorization mode.

Fix – Buildtime

Kubernetes

apiVersion: v1

kind: Pod

metadata:

creationTimestamp: null

labels:

component: kube-apiserver

tier: control-plane

name: kube-apiserver

namespace: kube-system

spec:

containers:

– command:

+ – kube-apiserver

+ – –authorization-mode=RBAC,Node

image: gcr.io/google_containers/kube-apiserver-amd64:v1.6.0

livenessProbe:

failureThreshold: 8

httpGet:

host: 127.0.0.1

path: /healthz

port: 6443

scheme: HTTPS

initialDelaySeconds: 15

timeoutSeconds: 15

name: kube-apiserver-should-pass

resources:

requests:

cpu: 250m

volumeMounts:

– mountPath: /etc/kubernetes/

name: k8s

readOnly: true

– mountPath: /etc/ssl/certs

name: certs

– mountPath: /etc/pki

name: pki

hostNetwork: true

volumes:

– hostPath:

path: /etc/kubernetes

name: k8s

– hostPath:

path: /etc/ssl/certs

name: certs

– hostPath:

path: /etc/pki

name: pki

Ensure the –authorization-mode argument includes Node

Remediation: Edit the API server pod specification file /etc/kubernetes/manifests/kube-apiserver.yaml on the master node and set the –authorization-mode parameter to a value that includes Node.

–authorization-mode=Node,RBAC

Audit:

/bin/ps -ef | grep kube-apiserver | grep -v grep

Expected result:

‘Node,RBAC’ has ‘Node’

Ensure that the –profiling argument is set to false

Remediation: Edit the API server pod specification file /etc/kubernetes/manifests/kube-apiserver.yaml on the master node and set the below parameter.

–profiling=false

Audit:

/bin/ps -ef | grep kube-apiserver | grep -v grep

Expected result:

‘false’ is equal to ‘false’

Fix all of the following violations that were found against the Kubelet:- Ensure the –anonymous-auth argument is set to false.

Remediation: If using a Kubelet config file, edit the file to set authentication: anonymous: enabled to false. If using executable arguments, edit the kubelet service file /etc/systemd/system/kubelet.service.d/10-kubeadm.conf on each worker node and set the below parameter in KUBELET_SYSTEM_PODS_ARGS variable.

–anonymous-auth=false

Based on your system, restart the kubelet service. For example:

systemctl daemon-reload

systemctl restart kubelet.service

Audit:

/bin/ps -fC kubelet

Audit Config:

/bin/cat /var/lib/kubelet/config.yaml

Expected result:

‘false’ is equal to ‘false’

2) Ensure that the –authorization-mode argument is set to Webhook.

Audit

docker inspect kubelet | jq -e ‘.[0].Args[] | match(“–authorization-mode=Webhook”).string’ Returned Value: –authorization-mode=Webhook Fix all of the following violations that were found against the ETCD:- a. Ensure that the –auto-tls argument is not set to true Do not use self-signed certificates for TLS. etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should not be available to unauthenticated clients. You should enable the client authentication via valid certificates to secure the access to the etcd service.

Fix – Buildtime

Kubernetes

apiVersion: v1

kind: Pod

metadata:

annotations:

scheduler.alpha.kubernetes.io/critical-pod: “”

creationTimestamp: null

labels:

component: etcd

tier: control-plane

name: etcd

namespace: kube-system

spec:

containers:

– command:

+ – etcd

+ – –auto-tls=true

image: k8s.gcr.io/etcd-amd64:3.2.18

imagePullPolicy: IfNotPresent

livenessProbe:

exec:

command:

– /bin/sh

– -ec

– ETCDCTL_API=3 etcdctl –endpoints=https://[192.168.22.9]:2379 –cacert=/etc/kubernetes/pki/etcd/ca.crt

–cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt –key=/etc/kubernetes/pki/etcd/healthcheck-client.key get foo failureThreshold: 8 initialDelaySeconds: 15 timeoutSeconds: 15 name: etcd-should-fail resources: {} volumeMounts:

– mountPath: /var/lib/etcd

name: etcd-data

– mountPath: /etc/kubernetes/pki/etcd

name: etcd-certs

hostNetwork: true

priorityClassName: system-cluster-critical

volumes:

– hostPath:

path: /var/lib/etcd

type: DirectoryOrCreate

name: etcd-data

– hostPath:

path: /etc/kubernetes/pki/etcd

type: DirectoryOrCreate

name: etcd-certs

status: {}

NO.33 Cluster: qa-cluster Master node: master Worker 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 named restricted-policy to restrict access to Pod product running in namespace dev. Only allow the following Pods to connect to Pod products-service: 1. Pods in the namespace qa 2. Pods with label environment: stage, in any namespace

```
candidate@cli:~$ kubectl config use-context KSSH00301
Switched to context "KSSH00301".
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ kubectl get ns dev-team --show-labels
NAME       STATUS   AGE     LABELS
dev-team   Active   6h39m   environment=dev,kubernetes.io/metadata.name=dev-team
candidate@cli:~$ kubectl get pods -n dev-team --show-labels
NAME            READY   STATUS    RESTARTS   AGE     LABELS
users-service   1/1     Running   0          6h40m   environment=dev
candidate@cli:~$ ls
KSCH00301  KSMV00102  KSSC00301  KSSH00401     test-secret-pod.yaml
KSCS00101  KSMV00301  KSSH00301  password.txt  username.txt
candidate@cli:~$ vim np.yaml
```

free exams4sures.com

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: pod-access
  namespace: dev-team
spec:
  podSelector:
    matchLabels:
      environment: dev
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              environment: dev
        - podSelector:
            matchLabels:
              environment: testing
```

e.exams4sures.co

exams4sures.com

ee.exams4sures.com

NO.34 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 namespace nato.

Look for images with High or Critical severity 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.

```
candidate@cli:~$ kubectl config use-context KSSC00401
Switched to context "KSSC00401".
candidate@cli:~$ ssh kssc00401-master
Warning: Permanently added '10.240.86.231' (ECDSA) to the list of known hosts.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@kssc00401-master:~# kubectl get pods -n naboo
NAME           READY   STATUS    RESTARTS   AGE
c-3po          1/1     Running   0          6h48m
chewbacca      1/1     Running   0          6h48m
jawas          1/1     Running   0          6h48m
qui-gon-jinn   1/1     Running   0          6h48m
root@kssc00401-master:~# kubectl get pods -n naboo -o name
pod/c-3po
pod/chewbacca
pod/jawas
pod/qui-gon-jinn
root@kssc00401-master:~# for i in $(kubectl get pods -n naboo -o name)
> do
> kubectl get ${i} -o yaml | grep -i image
> done
Error from server (NotFound): pods "c-3po" not found
Error from server (NotFound): pods "chewbacca" not found
Error from server (NotFound): pods "jawas" not found
Error from server (NotFound): pods "qui-gon-jinn" not found
root@kssc00401-master:~# for i in $(kubectl get pods -n naboo -o name); do kubectl -n naboo
get ${i} -o yaml | grep -i image ; done
    image: centos:centos7.9.2009
    imagePullPolicy: Never
    image: centos:centos7.9.2009
    imageID: docker-pullable://centos@sha256:c73f515d06b0fa07bb18d8202035e739a494ce760aa7312
9f60f4bf2bd22b407
    image: photon:3.0
    imagePullPolicy: Never
    image: photon:3.0
    imageID: docker-pullable://photon@sha256:c48d61f0f3ad19215b75e2087cfbe95d7321abb454e4295
a0e6c38f563ece622
    image: alpine:3.7
    imagePullPolicy: Never
    image: alpine:3.7
    imageID: docker-pullable://alpine@sha256:8421d9a84432575381bfabd248f1eb56f3aa21d9d7cd251
1583c68c9b7511d10
    image: amazonlinux:2
    imagePullPolicy: Never
    image: amazonlinux:2
    imageID: docker-pullable://amazonlinux@sha256:246ef631c75ea83005889621119fd5cc9cbb5500e1
93707c38b6c060d597a146
root@kssc00401-master:~# trivy image centos:centos7.9.2009
2022-05-20T15:39:51.733Z        INFO    Need to update DB
2022-05-20T15:39:51.733Z        INFO    Downloading DB...
27.97 MiB / 27.97 MiB [----------------------------------------] 100.00% 27.43 MiB p/s 1s
```

free exams4sures.com

```
root@kssc00401-master:~# for i in $(kubectl get pods -n naboo -o name); do kubectl -n naboo
get ${i} -o yaml | grep -i image ; done
    image: centos:centos7.9.2009
    imagePullPolicy: Never
    image: centos:centos7.9.2009
    imageID: docker-pullable://centos@sha256:c73f515d06b0fa07bb18d8202035e739a494ce760aa7312
9f60f4bf2bd22b407
    image: photon:3.0
    imagePullPolicy: Never
    image: photon:3.0
    imageID: docker-pullable://photon@sha256:c48d61f0f1ad[illegible]5e2087cfbe95d7321abb454e4295
a0e6c38f563ece622
    image: alpine:3.7
    imagePullPolicy: Never
    image: alpine:3.7
    imageID: docker-pullable://alpine@sha256:8421d9a84432575381bfabd248f1eb56f3aa21d9d7cd251
1583c68c9b7511d[illegible]
    image: amazonlinux:2
    imagePullPolicy: Never
    image: amazonlinux:2
    imageID: docker-pullable://amazonlinux@sha256:246ef631c75ea83005889621119fd5cc9cbb5500e1
93707c38b6c060d597a146
root@kssc00401-master:~# trivy image photon:3.0
2022-05-20T15:40:18.003Z        INFO    Detected OS: photon
2022-05-20T15:40:18.003Z        INFO    Detecting Photon Linux vulnerabilities...
2022-05-20T15:40:18.005Z        INFO    Number of language-specific files: 0

photon:3.0 (photon 3.0)
=======================
Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)
```

```
root@kssc00401-master:~# kubectl get pods -n naboo -o name
pod/c-3po
pod/chewbacca
pod/jawas
pod/qui-gon-jinn
root@kssc00401-master:~# kubectl -n naboo pod/c-3po -o yaml | grep image
Error: flags cannot be placed before plugin name: -n
root@kssc00401-master:~# kubectl -n naboo get pod/c-3po -o yaml | grep image
    image: centos:centos7.9.2009
    imagePullPolicy: Never
    image: centos:centos7.9.2009
    imageID: docker-pullable://centos@sha256:c73f515d06b0fa07bb18d8202035e739a494ce760aa7312
9f60f4bf2bd22b407
root@kssc00401-master:~# kubectl -n naboo delete pod/c-3po
pod "c-3po" deleted
root@kssc00401-master:~# kubectl -n naboo delete pod/jawas
pod "jawas" deleted
```

free.exams4sures.com

```
pod "jawas" deleted
root@kssc00401-master:~# history
    1  kubectl get pods -n naboo
    2  kubectl get pods -n naboo -o name
    3  for i in $(kubectl get pods -n naboo -o name); do kubectl get ${i} -o yaml | grep -i
image ; done
    4  for i in $(kubectl get pods -n naboo -o name); do kubectl -n naboo get ${i} -o yaml |
 grep -i image ; done
    5  trivy image centos:centos7.9.2009
    6  for i in $(kubectl get pods -n naboo -o name); do kubectl -n naboo get ${i} -o yaml |
 grep -i image ; done
    7  trivy image photon:3.0
    8  for i in $(kubectl get pods -n naboo -o name); do kubectl -n naboo get ${i} -o yaml |
 grep -i image ; done
    9  trivy image alpine:3.7
   10  for i in $(kubectl get pods -n naboo -o name); do kubectl -n naboo get ${i} -o yaml |
 grep -i image ; done
   11  trivy image amazonlinux:2
   12  kubectl get pods -n naboo -o name
   13  kubectl -n naboo pod/c-3po -o yaml | grep image
   14  kubectl -n naboo get pod/c-3po -o yaml | grep image
   15  kubectl -n naboo delete pod/c-3po
   16  kubectl -n naboo delete pod/jawas
   17  history
root@kssc00401-master:~#
```

free.exams4sures.com

NO.35 Service is running on port 389 inside the system, find the process-id of the process, and stores the names of all the open-files inside the /candidate/KH77539/files.txt, and also delete the binary.
root# netstat -ltnup

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name tcp 0 0 127.0.0.1:17600 0.0.0.0:* LISTEN 1293/dropbox tcp 0 0 127.0.0.1:17603 0.0.0.0:* LISTEN 1293/dropbox tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 575/sshd tcp 0 0 127.0.0.1:9393 0.0.0.0:* LISTEN 900/perl tcp 0 0 :::80 :::* LISTEN 9583/docker-proxy tcp 0 0 :::443 :::* LISTEN 9571/docker-proxy udp 0 0 0.0.0.0:68 0.0.0.0:* 8822/dhcpcd

…

root# netstat -ltnup | grep ‘:22’

tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 575/sshd

The ss command is the replacement of the netstat command.

Now let’s see how to use the ss command to see which process is listening on port 22:

root# ss -ltnup ‘sport = :22’

Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port

tcp LISTEN 0 128 0.0.0.0:22 0.0.0.0:* users:(“sshd”,pid=575,fd=3))

NO.36 Cluster: qa-cluster

Master node: master Worker 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 named restricted-policy to restrict access to Pod product running in namespace dev.

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

1. Pods in the namespace qa

2. Pods with label environment: stage, in any namespace
$ 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/

NO.37 You can switch the cluster/configuration context using the following command: [desk@cli] $ kubectl config use-context qa Context: A pod fails to run because of an incorrectly specified ServiceAccount Task: Create a new service account named backend-qa in an existing namespace qa, which must not have access to any secret. Edit the frontend pod yaml to use backend-qa service account Note: You can find the frontend pod yaml at /home/cert_masters/frontend-pod.yaml
[desk@cli] $ k create sa backend-qa -n qa sa/backend-qa created [desk@cli] $ k get role,rolebinding -n qa No resources found in qa namespace. [desk@cli] $ k create role backend -n qa –resource pods,namespaces,configmaps –verb list # No access to secret [desk@cli] $ k create rolebinding backend -n qa –role backend –serviceaccount qa:backend-qa [desk@cli] $ vim /home/cert_masters/frontend-pod.yaml apiVersion: v1 kind: Pod metadata:

name: frontend

spec:

serviceAccountName: backend-qa # Add this

image: nginx

name: frontend

[desk@cli] $ k apply -f /home/cert_masters/frontend-pod.yaml pod created

[desk@cli] $ k create sa backend-qa -n qa serviceaccount/backend-qa created [desk@cli] $ k get role,rolebinding -n qa No resources found in qa namespace. [desk@cli] $ k create role backend -n qa –resource pods,namespaces,configmaps –verb list role.rbac.authorization.k8s.io/backend created [desk@cli] $ k create rolebinding backend -n qa –role backend –serviceaccount qa:backend-qa rolebinding.rbac.authorization.k8s.io/backend created [desk@cli] $ vim

/home/cert_masters/frontend-pod.yaml apiVersion: v1 kind: Pod metadata:

name: frontend

spec:

serviceAccountName: backend-qa # Add this

image: nginx

name: frontend

[desk@cli] $ k apply -f /home/cert_masters/frontend-pod.yaml pod/frontend created
https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/

NO.38 SIMULATION

Before Making any changes build the Dockerfile with tag base:v1

Now Analyze and edit the given Dockerfile(based on ubuntu 16:04)

Fixing two instructions present in the file, Check from Security Aspect and Reduce Size point of view.

Dockerfile:

FROM ubuntu:latest

RUN apt-get update -y

RUN apt install nginx -y

COPY entrypoint.sh /

RUN useradd ubuntu

ENTRYPOINT [“/entrypoint.sh”]

USER ubuntu

entrypoint.sh

#!/bin/bash

echo “Hello from CKS”

After fixing the Dockerfile, build the docker-image with the tag base:v2 To Verify: Check the size of the image before and after the build.
* Send us the Feedback on it.

NO.39 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 named web-pod running in the namespace database. 1. Edit the existing Role bound to the Pod’s ServiceAccount test-sa to only allow performing get operations, only on resources of type Pods. 2. Create a new Role named test-role-2 in the namespace database, which only allows performing update operations, only on resources of type statuefulsets. 3. Create a new RoleBinding named test-role-2-bind binding the newly created Role to the Pod’s ServiceAccount. Note: Don’t delete the existing RoleBinding.

```
candidate@cli:~$ kubectl config use-context KSCH00201
Switched to context "KSCH00201".
candidate@cli:~$ kubectl get pods -n security
NAME      READY   STATUS    RESTARTS   AGE
web-pod   1/1     Running   0          6h9m
candidate@cli:~$ kubectl get deployments.apps -n security
No resources found in security namespace.
candidate@cli:~$ kubectl describe rolebindings.rbac.authorization.k8s.io -n security
Name:         dev-role
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  dev-role
Subjects:
  Kind           Name      Namespace
  ----           ----      ---------
  ServiceAccount  sa-dev-1
candidate@cli:~$ kubectl describe role dev-role -n security
Name:         dev-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  *          []                 []              [*]
candidate@cli:~$ kubectl edit role/dev-role -n security
```

free.exams4sures.com

```
  uid: b4c9ddd6-2729-43bd-8fbd-b2d227f4c4cd
rules:
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - watch
```

free.exams4sures.com

```
candidate@cli:~$ kubectl describe role dev-role -n security
Name:         dev-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  *          []                 []              [*]
candidate@cli:~$ kubectl edit role/dev-role -n security
role.rbac.authorization.k8s.io/dev-role edited
candidate@cli:~$ kubectl describe role dev-role -n security
Name:         dev-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  services   []                 []              [watch]
candidate@cli:~$ kubectl get pods -n security
NAME      READY   STATUS    RESTARTS   AGE
web-pod   1/1     Running   0          6h12m
candidate@cli:~$ kubectl get pods/web-pod -n security -o yaml | grep serviceAccount
  serviceAccount: sa-dev-1
  serviceAccountName: sa-dev-1
      - serviceAccountToken:
candidate@cli:~$ kubectl create role role-2 --verb=update --resource=namespaces -n security
role.rbac.authorization.k8s.io/role-2 created
candidate@cli:~$ kubectl create rolebinding role-2-binding --role
--role   --role=
candidate@cli:~$ kubectl create rolebinding role-2-binding --role=role-2 --serviceaccount=se
curity:sa-dev-1 -n security
rolebinding.rbac.authorization.k8s.io/role-2-binding created
candidate@cli:~$
```

free.exams4sures.com

NO.40 SIMULATION

Analyze and edit the given Dockerfile

FROM ubuntu:latest

RUN apt-get update -y

RUN apt-install nginx -y

COPY entrypoint.sh /

ENTRYPOINT [“/entrypoint.sh”]

USER ROOT

Fixing two instructions present in the file being prominent security best practice issues Analyze and edit the deployment manifest file apiVersion: v1 kind: Pod metadata:

name: security-context-demo-2

spec:

securityContext:

runAsUser: 1000

containers:

– name: sec-ctx-demo-2

image: gcr.io/google-samples/node-hello:1.0

securityContext:

runAsUser: 0

privileged: True

allowPrivilegeEscalation: false

Fixing two fields present in the file being prominent security best practice issues Don’t add or remove configuration settings; only modify the existing configuration settings Whenever you need an unprivileged user for any of the tasks, use user test-user with the user id 5487
* Send us the Feedback on it.

NO.41 Before Making any changes build the Dockerfile with tag base:v1

Now Analyze and edit the given Dockerfile(based on ubuntu 16:04)

Fixing two instructions present in the file, Check from Security Aspect and Reduce Size point of view.

Dockerfile:

FROM ubuntu:latest

RUN apt-get update -y

RUN apt install nginx -y

COPY entrypoint.sh /

RUN useradd ubuntu

ENTRYPOINT [“/entrypoint.sh”]

USER ubuntu

entrypoint.sh

#!/bin/bash

echo “Hello from CKS”

After fixing the Dockerfile, build the docker-image with the tag base:v2
* To Verify: Check the size of the image before and after the build.

NO.42 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 named adam in the safe namespace.

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 names newsecret in the safe namespace, with the following content:

Username: dbadmin

Password: moresecurepas

Finally, create a new Pod that has access to the secret newsecret via a volume:

Namespace: safe

Pod name: mysecret-pod

Container name: db-container

Image: redis

Volume name: secret-vol

Mount path: /etc/mysecret
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"
  selfLink: /api/v1/namespaces/safe/secrets/adam
  uid: da04fa1e-22a5-49d0-95d5-d73e40542033
type: Opaque
```

free.exams4sures.com

```
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 $
```

NO.43 On the Cluster worker node, enforce the prepared AppArmor profile

#include <tunables/global>

profile nginx-deny flags=(attach_disconnected) {

#include <abstractions/base>

file,

Deny all file writes.

deny /** w,

}

EOF’
* Edit the prepared manifest file to include the AppArmor profile.
apiVersion: v1

kind: Pod

metadata:

name: apparmor-pod

spec:

containers:

– name: apparmor-pod

image: nginx

Finally, apply the manifests files and create the Pod specified on it.

Verify: Try to make a file inside the directory which is restricted.

NO.44 Create a new ServiceAccount named backend-sa in the existing namespace default, which has the capability to list the pods inside the namespace default.

Create a new Pod named backend-pod in the namespace default, mount the newly created sa backend-sa to the pod, and Verify that the pod is able to list pods.

Ensure that the Pod is running.
A service account provides an identity for processes that run in a Pod.

When you (a human) access the cluster (for example, using kubectl), you are authenticated by the apiserver as a particular User Account (currently this is usually admin, unless your cluster administrator has customized your cluster). Processes in containers inside pods can also contact the apiserver. When they do, they are authenticated as a particular Service Account (for example, default).

When you create a pod, if you do not specify a service account, it is automatically assigned the default service account in the same namespace. If you get the raw json or yaml for a pod you have created (for example, kubectl get pods/<podname> -o yaml), you can see the spec.serviceAccountName field has been automatically set.

You can access the API from inside a pod using automatically mounted service account credentials, as described in Accessing the Cluster. The API permissions of the service account depend on the authorization plugin and policy in use.

In version 1.6+, you can opt out of automounting API credentials for a service account by setting automountServiceAccountToken: false on the service account:

apiVersion: v1

kind: ServiceAccount

metadata:

name: build-robot

automountServiceAccountToken: false

…

In version 1.6+, you can also opt out of automounting API credentials for a particular pod:

apiVersion: v1

kind: Pod

metadata:

name: my-pod

spec:

serviceAccountName: build-robot

automountServiceAccountToken: false

…

The pod spec takes precedence over the service account if both specify a automountServiceAccountToken value.

NO.45 a. Retrieve the content of the existing secret named default-token-xxxxx in the testing namespace.

Store the value of the token in the token.txt

b. Create a new secret named test-db-secret in the DB namespace with the following content:

username: mysql

password: password@123

Create the Pod name test-db-pod of image nginx in the namespace db that can access test-db-secret via a volume at path /etc/mysql-credentials
To add a Kubernetes cluster to your project, group, or instance:

Navigate to your:

Project’s Operations > Kubernetes page, for a project-level cluster.

Group’s Kubernetes page, for a group-level cluster.

Admin Area > Kubernetes page, for an instance-level cluster.

Click Add Kubernetes cluster.

Click the Add existing cluster tab and fill in the details:

Kubernetes cluster name (required) – The name you wish to give the cluster.

Environment scope (required) – The associated environment to this cluster.

API URL (required) – It’s the URL that GitLab uses to access the Kubernetes API. Kubernetes exposes several APIs, we want the “base” URL that is common to all of them. For example, https://kubernetes.example.com rather than https://kubernetes.example.com/api/v1.

Get the API URL by running this command:

kubectl cluster-info | grep -E ‘Kubernetes master|Kubernetes control plane’ | awk ‘/http/ {print $NF}’ CA certificate (required) – A valid Kubernetes certificate is needed to authenticate to the cluster. We use the certificate created by default.

List the secrets with kubectl get secrets, and one should be named similar to default-token-xxxxx. Copy that token name for use below.

Get the certificate by running this command:

kubectl get secret <secret name> -o jsonpath=”{[‘data’][‘ca.crt’]}”

NO.46 A container image scanner is set up on the cluster.

Given an incomplete configuration in the directory

/etc/kubernetes/confcontrol and a functional container image scanner with HTTPS endpoint https://test-server.local.8081/image_policy
* 1. Enable the admission plugin.
2. Validate the control configuration and change it to implicit deny.

Finally, test the configuration by deploying the pod having the image tag as latest.

NO.47 SIMULATION

Using the runtime detection tool Falco, Analyse the container behavior for at least 30 seconds, using filters that detect newly spawning and executing processes store the incident file art /opt/falco-incident.txt, containing the detected incidents. one per line, in the format

[timestamp],[uid],[user-name],[processName]
* Sendusyoursuggestiononit

NO.48 Task

Create a NetworkPolicy named pod-access to restrict access to Pod users-service running in namespace dev-team.

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

```
candidate@cli:~$ kubectl config use-context KSSH00301
Switched to context "KSSH00301".
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ kubectl get ns dev-team --show-labels
NAME       STATUS   AGE     LABELS
dev-team   Active   6h39m   environment=dev,kubernetes.io/metadata.name=dev-team
candidate@cli:~$ kubectl get pods -n dev-team --show-labels
NAME            READY   STATUS    RESTARTS   AGE     LABELS
users-service   1/1     Running   0          6h40m   environment=dev
candidate@cli:~$ ls
KSCH00301  KSMV00102  KSSC00301  KSSH00401     test-secret-pod.yaml
KSCS00101  KSMV00301  KSSH00301  password.txt  username.txt
candidate@cli:~$ vim np.yaml
```

free.exams4sures.com

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: pod-access
  namespace: dev-team
spec:
  podSelector:
    matchLabels:
      environment: dev
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              environment: dev
        - podSelector:
            matchLabels:
              environment: testing
```

e.exams4sures.co

```
candidate@cli:~$ vim np.yaml
candidate@cli:~$ cat np.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: pod-access
  namespace: dev-team
spec:
  podSelector:
    matchLabels:
      environment: dev
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              environment: dev
        - podSelector:
            matchLabels:
              environment: testing
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ kubectl create -f np.yaml -n dev-team
networkpolicy.networking.k8s.io/pod-access created
candidate@cli:~$ kubectl describe netpol -n dev-team
Name:         pod-access
Namespace:    dev-team
Created on:   2022-05-20 15:35:33 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     environment=dev
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From:
      NamespaceSelector: environment=dev
    From:
      PodSelector: environment=testing
  Not affecting egress traffic
  Policy Types: Ingress
candidate@cli:~$ cat KSSH00301/network-policy.yaml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ""
  namespace: ""
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from: []
    - from: []
candidate@cli:~$ cp np.yaml KSSH00301/network-policy.yaml
candidate@cli:~$ cat KSSH00301/network-policy.yaml
```

ee.exams4sures.co

NO.49 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 named not-trusted using the prepared runtime handler names runsc.

Update all Pods in the namespace server to run on newruntime.
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

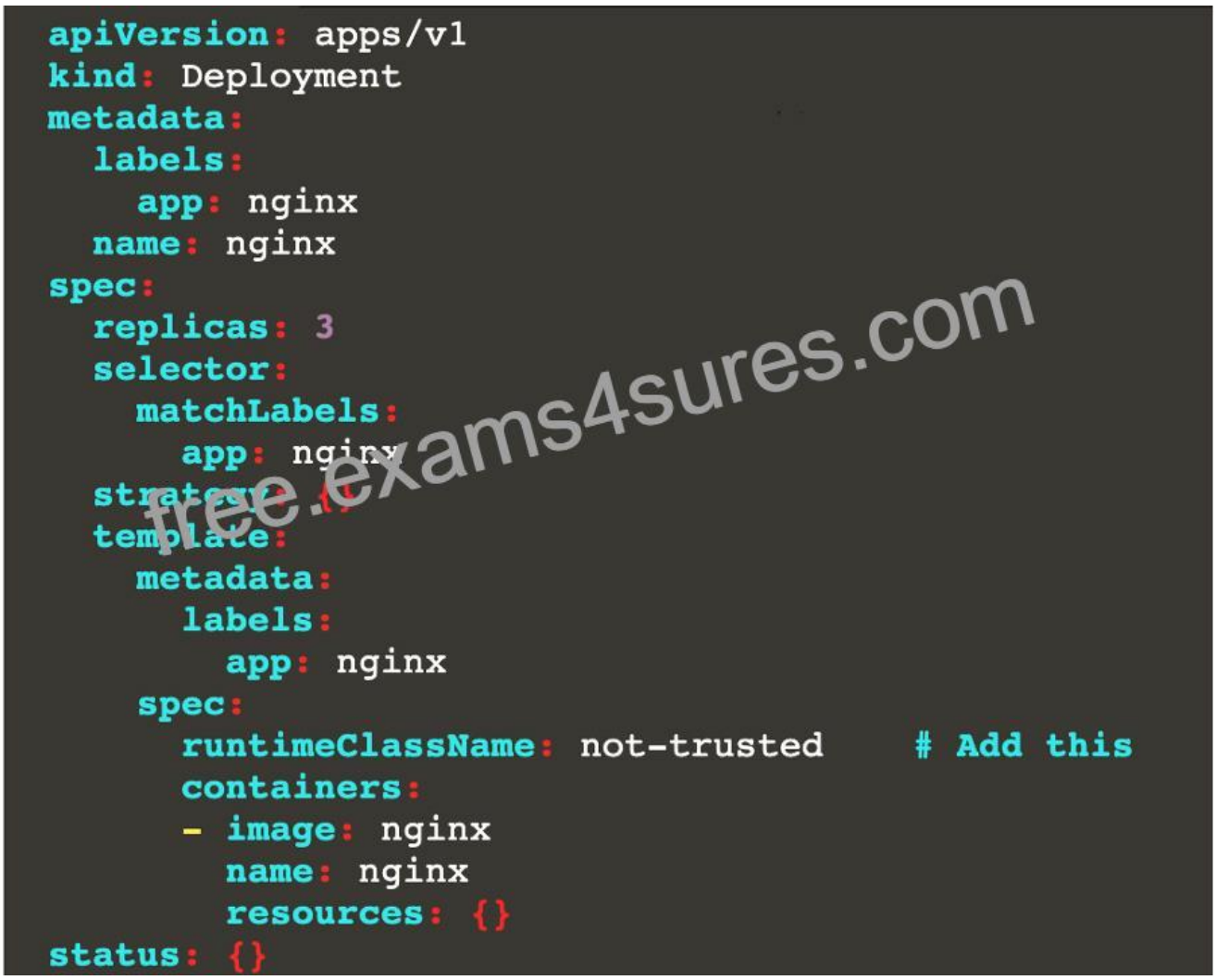
nginx-6798fc88e8-ndved 1/1 Running 0 11m

[desk@cli] $ k get deploy

NAME READY UP-TO-DATE AVAILABLE AGE

nginx 3/3 11 3 5m

[desk@cli] $ k edit deploy nginx



free.exams4sures.com

NO.50 Create a network policy named restrict-np to restrict to pod nginx-test running in namespace testing.

Only allow the following Pods to connect to Pod nginx-test:-

1. pods in the namespace default

2. pods with label version:v1 in any namespace.

Make sure to apply the network policy.
* Send us your Feedback on this.

NO.51 You can switch the cluster/configuration context using the following command: [desk@cli] $ kubectl config use-context test-account Task: Enable audit logs in the cluster.

To do so, enable the log backend, and ensure that:

1. logs are stored at /var/log/Kubernetes/logs.txt

2. log files are retained for 5 days

3. at maximum, a number of 10 old audit log files are retained

A basic policy is provided at /etc/Kubernetes/logpolicy/audit-policy.yaml. It only specifies what not to log. Note: The base policy is located on the cluster’s master node.

Edit and extend the basic policy to log: 1. Nodes changes at RequestResponse level 2. The request body of persistentvolumes changes in the namespace frontend 3. ConfigMap and Secret changes in all namespaces at the Metadata level Also, add a catch-all rule to log all other requests at the Metadata level Note: Don’t forget to apply the modified policy.
$ vim /etc/kubernetes/log-policy/audit-policy.yaml

– level: RequestResponse

userGroups: [“system:nodes”]

– level: Request

resources:

– group: “” # core API group

resources: [“persistentvolumes”]

namespaces: [“frontend”]

– level: Metadata

resources:

– group: “”

resources: [“configmaps”, “secrets”]

– level: Metadata

$ vim /etc/kubernetes/manifests/kube-apiserver.yaml Add these

– –audit-policy-file=/etc/kubernetes/log-policy/audit-policy.yaml

– –audit-log-path=/var/log/kubernetes/logs.txt

– –audit-log-maxage=5

– –audit-log-maxbackup=10

Explanation

[desk@cli] $ ssh master1 [master1@cli] $ vim /etc/kubernetes/log-policy/audit-policy.yaml apiVersion: audit.k8s.io/v1 # This is required.

kind: Policy

Don’t generate audit events for all requests in RequestReceived stage.

omitStages:

– “RequestReceived”

rules:

Don’t log watch requests by the “system:kube-proxy” on endpoints or services

– level: None

users: [“system:kube-proxy”]

verbs: [“watch”]

resources:

– group: “” # core API group

resources: [“endpoints”, “services”]

Don’t log authenticated requests to certain non-resource URL paths.

– level: None

userGroups: [“system:authenticated”]

nonResourceURLs:

– “/api*” # Wildcard matching.

– “/version”

Add your changes below

– level: RequestResponse

userGroups: [“system:nodes”] # Block for nodes

– level: Request

resources:

– group: “” # core API group

resources: [“persistentvolumes”] # Block for persistentvolumes

namespaces: [“frontend”] # Block for persistentvolumes of frontend ns

– level: Metadata

resources:

– group: “” # core API group

resources: [“configmaps”, “secrets”] # Block for configmaps & secrets

– level: Metadata # Block for everything else

[master1@cli] $ vim /etc/kubernetes/manifests/kube-apiserver.yaml

apiVersion: v1

kind: Pod

metadata:

annotations:

kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 10.0.0.5:6443 labels:

component: kube-apiserver

tier: control-plane

name: kube-apiserver

namespace: kube-system

spec:

containers:

– command:

– kube-apiserver

– –advertise-address=10.0.0.5

– –allow-privileged=true

– –authorization-mode=Node,RBAC

– –audit-policy-file=/etc/kubernetes/log-policy/audit-policy.yaml #Add this

– –audit-log-path=/var/log/kubernetes/logs.txt #Add this

– –audit-log-maxage=5 #Add this

– –audit-log-maxbackup=10 #Add this

…

output truncated

Note: log volume & policy volume is already mounted in vim /etc/kubernetes/manifests/kube-apiserver.yaml so no need to mount it.
Reference: https://kubernetes.io/docs/tasks/debug-application-cluster/audit/

NO.52 use the Trivy to scan the following images,
* 1. amazonlinux:1
2. k8s.gcr.io/kube-controller-manager:v1.18.6

Look for images with HIGH or CRITICAL severity vulnerabilities and store the output of the same in /opt/trivy-vulnerable.txt

NO.53 SIMULATION

Create a network policy named restrict-np to restrict to pod nginx-test running in namespace testing.

Only allow the following Pods to connect to Pod nginx-test:-

1. pods in the namespace default

2. pods with label version:v1 in any namespace.

Make sure to apply the network policy.
* Send us your Feedback on this.

Linux Foundation CKS (Certified Kubernetes Security Specialist) Exam is a certification program designed to test and validate the knowledge and skills of professionals in Kubernetes security. Kubernetes is an open-source container orchestration platform that is widely used by organizations to manage their containerized applications. As Kubernetes grows in popularity, the need for professionals with expertise in securing Kubernetes environments has also increased.

Linux Foundation Exam 2024 CKS Dumps Updated Questions:
https://www.exams4sures.com/Linux-Foundation/CKS-practice-exam-dumps.html]