CKS Q&As

Certified Kubernetes Security Specialist (CKS) Exam

Pass Linux Foundation CKS Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.leads4pass.com/cks.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Linux Foundation Official Exam Center

QUESTION 1

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 the ubuntu:18:04 image)

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

Analyse and edit the given manifest file

/home/cert_masters/mydeployment.yaml fixing 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 user nobody with user id 65535

A. See the explanation below

B. PlaceHolder

Correct Answer: A

1. For Dockerfile: Fix the image version and user name in Dockerfile2. 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"]
```

Text

[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

QUESTION 2

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

A. See the below.

B. PlaceHolder

Correct Answer: A

QUESTION 3

Create a PSP that will prevent the creation of privileged pods in the namespace.

Create a new PodSecurityPolicy named prevent-privileged-policy which prevents the creation of privileged pods.

Create a new ServiceAccount named psp-sa in the namespace default.

Create a new ClusterRole named prevent-role, which uses the newly created Pod Security Policy prevent-privileged-policy.

Create a new ClusterRoleBinding named prevent-role-binding, which binds the created ClusterRole prevent-role to the created SA psp-sa.

Also, Check the Configuration is working or not by trying to Create a Privileged pod, it should get failed.

A. See the below.

B. PlaceHolder

Correct Answer: A

Create a PSP that will prevent the creation of privileged pods in the namespace. $ cat clusterrole-use-privileged.yaml apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole metadata: name: use-privileged-psp rules:

-apiGroups: [\\'policy\\']

resources: [\\'podsecuritypolicies\\']

verbs: [\\'use\\']

resourceNames:

-default-psp

apiVersion: rbac.authorization.k8s.io/v1 kind: RoleBinding metadata: name: privileged-role-bind namespace: psp-test roleRef: apiGroup: rbac.authorization.k8s.io kind: ClusterRole name: use-privileged-psp subjects:

-kind: ServiceAccount name: privileged-sa $ kubectl -n psp-test apply -f clusterrole-use-privileged.yaml

After a few moments, the privileged Pod should be created.

Create a new PodSecurityPolicy named prevent-privileged-policy which prevents the creation of privileged pods.

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

metadata:

name: example

spec:

privileged: false # Don\\'t allow privileged pods!

The rest fills in some required fields.

seLinux:

rule: RunAsAny

supplementalGroups:

rule: RunAsAny

runAsUser:

rule: RunAsAny

fsGroup:

rule: RunAsAny

volumes:

-\\'*\\'

And create it with kubectl:

kubectl-admin create -f example-psp.yaml

Now, as the unprivileged user, try to create a simple pod:

kubectl-user create -f-