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.passapply.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

Task Analyze and edit the given Dockerfile /home/candidate/KSSC00301/Docker file (based on the ubuntu:16.04 image), fixing two instructions present in the file that are prominent security/best-practice issues. Analyze and edit the given manifest file /home/candidate/KSSC00301/deployment.yaml, fixing two fields present in the file that are prominent security/best-practice issues.

You **must** complete this task on the following cluster/nodes:

Cluster	Master node	Worker node
KSSC00301	kssc00301-master	kssc00301-worker1

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

```
[candidate@cli] $ kubectl config use-context KSSC00301
```

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 explanation below.

B. PlaceHolder

Correct Answer: A

QUESTION 2

CORRECT TEXT Your organization\\'s security policy includes:

You **must** complete this task on the following cluster/nodes:

Cluster	Master node	Worker node
KSCH00301	ksch00301-master	ksch00301-worker1

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

```
[candidate@cli] $ kubectl config use-context KSCH00301
```

1.

ServiceAccounts must not automount API credentials

2.

ServiceAccount names must end in "-sa"

The Pod specified in the manifest file /home/candidate/KSCH00301 /pod-m

nifest.yaml fails to schedule because of an incorrectly specified ServiceAccount.

Complete the following tasks:

Task

1.

Create a new ServiceAccount named frontend-sa in the existing namespace qa. Ensure the ServiceAccount does not automount API credentials.

2.

Using the manifest file at /home/candidate/KSCH00301 /pod-manifest.yaml, create the Pod.

3.

Finally, clean up any unused ServiceAccounts in namespace qa.

A. See the explanation below

B. PlaceHolder

Correct Answer: A

QUESTION 3

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.

A. See explanation below.

B. PlaceHolder

Correct Answer: A

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))

QUESTION 4

```
candidate@cli:~$ kubectl config use-context KSCS00101
Switched to context "KSCS00101".
candidate@cli:~$ cat /home/candidate/KSCS00101/network-policy.yaml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ""
  namespace: ""
spec:
  podSelector: {}
  policyTypes: []
candidate@cli:~$ vim /home/candidate/KSCS00101/network-policy.yaml
candidate@cli:~$
```

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: "defaultdeny"
  namespace: "testing"
spec:
  podSelector: {}
  policyTypes:
  - Egress
  egress:
  - to:
    - podSelector: {}
      namespaceSelector:
        matchLabels:
          access: testingproject
```

```
candidate@cli:~$ vim /home/candidate/KSCS00101/network-policy.yaml
candidate@cli:~$ vim /home/candidate/KSCS00101/network-policy.yaml
candidate@cli:~$ kubectl label ns testing access=testingproject
namespace/testing labeled
candidate@cli:~$ cat /home/candidate/KSCS00101/network-policy.yaml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: "defaultdeny"
  namespace: "testing"
spec:
  podSelector: {}
  policyTypes:
  - Egress
  egress:
  - to:
    - podSelector: {}
      namespaceSelector:
        matchLabels:
          access: testingproject
candidate@cli:~$ kubectl create -f /home/candidate/KSCS00101/network-policy.yaml
networkpolicy.networking.k8s.io/defaultdeny created
candidate@cli:~$ kubectl -n testing describe networkpolicy
Name:         defaultdeny
Namespace:    testing
Created on:   2022-05-20 14:28:27 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     <none> (Allowing the specific traffic to all pods in this namespace)
  Not affecting ingress traffic
  Allowing egress traffic:
    To Port: <any> (traffic allowed to all ports)
    To:
      NamespaceSelector: access=testingproject
      PodSelector: <none>
  Policy Types: Egress
candidate@cli:~$
```

Create a RuntimeClass named gvisor-rc using the prepared runtime handler named runsc.

Create a Pods of image Nginx in the Namespace server to run on the gVisor runtime class

A. See the explanation below:

B. PlaceHolder

Correct Answer: A

Install the Runtime Class for gVisor

{ # Step 1: Install a RuntimeClass cat