

## CKS Dumps

### Certified Kubernetes Security Specialist (CKS) Exam

<https://www.certleader.com/CKS-dumps.html>



**NEW QUESTION 1**

Create a new NetworkPolicy named deny-all in the namespace testing which denies all traffic of type ingress and egress traffic

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

You can create a "default" isolation policy for a namespace by creating a NetworkPolicy that selects all pods but does not allow any ingress traffic to those pods.

--

```
apiVersion: networking.k8s.io/v1
```

```
kind: NetworkPolicy
```

```
metadata:
```

```
name: default-deny-ingress
```

```
spec:
```

```
podSelector: {}
```

```
policyTypes:
```

```
- Ingress
```

You can create a "default" egress isolation policy for a namespace by creating a NetworkPolicy that selects all pods but does not allow any egress traffic from those pods.

--

```
apiVersion: networking.k8s.io/v1
```

```
kind: NetworkPolicy
```

```
metadata:
```

```
name: allow-all-egress
```

```
spec:
```

```
podSelector: {}
```

```
egress:
```

```
- {}
```

```
policyTypes:
```

```
- Egress
```

Default deny all ingress and all egress trafficYou can create a "default" policy for a namespace which prevents all ingress AND egress traffic by creating the following NetworkPolicy in that namespace.

--

```
apiVersion: networking.k8s.io/v1
```

```
kind: NetworkPolicy
```

```
metadata:
```

```
name: default-deny-all
```

```
spec:
```

```
podSelector: {}
```

```
policyTypes:
```

```
- Ingress
```

```
- Egress
```

This ensures that even pods that aren't selected by any other NetworkPolicy will not be allowed ingress or egress traffic.

**NEW QUESTION 2**

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](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.

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Send us your feedback on it.

**NEW QUESTION 3**

Enable audit logs in the cluster, To Do so, enable the log backend, and ensure that-

\* 1. logs are stored at /var/log/kubernetes/kubernetes-logs.txt.

\* 2. Log files are retainedfor5 days.

\* 3. at maximum, a number of 10 old audit logs files are retained.

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Edit and extend the basic policy to log:

\* 1. Cronjobs changes at RequestResponse

\* 2. Log the request body of deployments changesinthenamespacekube-system.

\* 3. Log all other resourcesincoreandextensions at the Request level.

\* 4. Don't log watch requests by the "system:kube-proxy" on endpoints or Send us your feedback on it.

**NEW QUESTION 4**

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 that the RotateKubeletServerCertificate argumentissettotrue.
- \* b. Ensure that the admission control plugin PodSecurityPolicyisset.
- \* c. Ensure that the --kubelet-certificate-authority argumentissetasappropriate.

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

- \* a. Ensure the --anonymous-auth argumentissettofalse.
- \* b. Ensure that the --authorization-mode argumentissetto Webhook.

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

- \* a. Ensure that the --auto-tls argumentisnotsettotrue
- \* b. Ensure that the --peer-auto-tls argumentisnotsettotrue

Hint: Take the use of Tool Kube-Bench

- A. Mastered
- B. Not Mastered

**Answer: A**

**Explanation:**

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

- \* a. Ensure that the RotateKubeletServerCertificate argumentissettotrue.

apiVersion: v1

kind: Pod

metadata:

creationTimestamp: null

labels:

component: kubelet

tier: control-plane

name: kubelet

namespace: kube-system

spec:

containers:

- command:

- kube-controller-manager

+ - --feature-gates=RotateKubeletServerCertificate=true

image: gcr.io/google\_containers/kubelet-amd64:v1.6.0

livenessProbe:

failureThreshold: 8

httpGet:

host: 127.0.0.1

path: /healthz

port: 6443

scheme: HTTPS

initialDelaySeconds: 15

timeoutSeconds: 15

name: kubelet

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

- \* b. Ensure that the admission control plugin PodSecurityPolicyisset.

audit: "/bin/ps -ef | grep \$apiserverbin | grep -v grep"

tests:

test\_items:

- flag: "--enable-admission-plugins"

compare:

op: has

value: "PodSecurityPolicy"

set: true

remediation: |

Follow the documentation and create Pod Security Policy objects as per your environment.

Then, edit the API server pod specification file \$apiserverconf

on the master node and set the --enable-admission-plugins parameter to a value that includes PodSecurityPolicy :

--enable-admission-plugins=...,PodSecurityPolicy,...

Then restart the API Server.

scored: true

\* c. Ensure that the --kubelet-certificate-authority argument is set as appropriate.

audit: "/bin/ps -ef | grep \$apiserverbin | grep -v grep"

tests:

test\_items:

- flag: "--kubelet-certificate-authority"

set: true

remediation: |

Follow the Kubernetes documentation and setup the TLS connection between the apiserver and kubelets. Then, edit the API server pod specification file

\$apiserverconf on the master node and set the --kubelet-certificate-authority parameter to the path to the cert file for the certificate authority.

--kubelet-certificate-authority=<ca-string>

scored: true

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

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

Edit the etcd pod specification file \$etcdconf on the master node and either remove the --auto-tls parameter or set it to false.--auto-tls=false

\* b. Ensure that the --peer-auto-tls argument is not set to true

Edit the etcd pod specification file \$etcdconf on the master node and either remove the --peer-auto-tls parameter or set it to false.--peer-auto-tls=false

## NEW QUESTION 5

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. Mastered

B. Not Mastered

**Answer: A**

### Explanation:

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.

## NEW QUESTION 6

On the Cluster worker node, enforce the prepared AppArmor profile

```
#include<tunables/global>
```

```
profile docker-nginx flags=(attach_disconnected,mediate_deleted) {
```

```
#include<abstractions/base>
```

```
network inet tcp,
```

```
network inet udp,
```

```
network inet icmp,
```

```
deny network raw,
```

```
deny network packet,
```

```
file,
```

```
umount,
```

```
deny /bin/** wl,
```

```
deny /boot/** wl,
```

```
deny /dev/** wl,
```

```
deny /etc/** wl,
```

```
deny /home/** wl,
```

```
deny /lib/** wl,
```

```
deny /lib64/** wl,
```

```
deny /media/** wl,
```

```
deny /mnt/** wl,
```

```
deny /opt/** wl,
```

```
deny /proc/** wl,
```

```
deny /root/** wl,
```

```
deny /sbin/** wl,
```

```
deny /srv/** wl,
```

```
deny /tmp/** wl,
```

```
deny /sys/** wl,
deny /usr/** wl,
audit /** w,
/var/run/nginx.pid w,
/usr/sbin/nginx ix,
deny /bin/dash mrwklx,
deny /bin/sh mrwklx,
deny /usr/bin/top mrwklx,
capability chown,
capability dac_override,
capability setuid,
capability setgid,
capability net_bind_service,
deny @{PROC}/* w, # deny write for all files directly in /proc (not in a subdir)
# deny write to files not in /proc/<number>/** or /proc/sys/**
deny @{PROC}/[^[1-9],[^1-9][^0-9],[^1-9s][^0-9y][^0-9s],[^1-9][^0-9][^0-9][^0-9]*/** w,
deny @{PROC}/sys/[^k]** w, # deny /proc/sys except /proc/sys/k* (effectively /proc/sys/kernel)
deny @{PROC}/sys/kernel/{?,??,[^s][^h][^m]**} w, # deny everything except shm* in
/proc/sys/kernel/
deny @{PROC}/sysrq-trigger rwklx,
deny @{PROC}/mem rwklx,
deny @{PROC}/kmem rwklx,
deny @{PROC}/kcore rwklx,
deny mount,
deny /sys/[^f]** wklx,
deny /sys/f[^s]** wklx,
deny /sys/fs/[^c]** wklx,
deny /sys/fs/c[^g]** wklx,
deny /sys/fs/cg[^r]** wklx,
deny /sys/firmware/** rwklx,
deny /sys/kernel/security/** rwklx,
}
```

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 use command ping, top, sh

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Send us your feedback on it.

**NEW QUESTION 7**

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]

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Send us your suggestion on it.

**NEW QUESTION 8**

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. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

```
Install the Runtime Class for gVisor
{ # Step 1: Install a RuntimeClass
cat <<EOF | kubectl apply -f -
apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:
  name: gvisor
```

```
handler: runsc
EOF
}
Create a Pod with the gVisor Runtime Class
{ # Step 2: Create a pod
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
name: nginx-gvisor
spec:
runtimeClassName: gvisor
containers:
- name: nginx
image: nginx
EOF
}
Verify that the Pod is running
{ # Step 3: Get the pod
kubectl get pod nginx-gvisor -o wide
}
```

**NEW QUESTION 9**

Create a Pod name Nginx-pod inside the namespace testing, Create a service for the Nginx-pod named nginx-svc, using the ingress of your choice, run the ingress on tls, secure port.

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Send us your feedback on it.

**NEW QUESTION 10**

Analyze and edit the given Dockerfile

```
FROM ubuntu:latest
RUN apt-getupdate -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

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

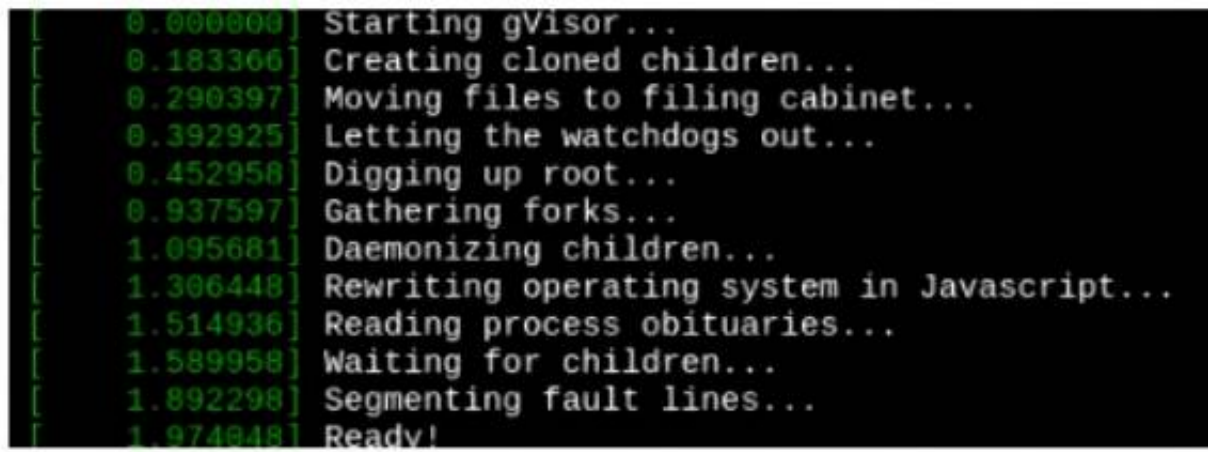
Whenever you need an unprivileged user for any of the tasks, use user test-user with the user id 5487 Send us the Feedback on it.

**NEW QUESTION 10**

Create a RuntimeClass named untrusted using the prepared runtime handler named runsc.

Create a Pods of image alpine:3.13.2 in the Namespace default to run on the gVisor runtime class. Verify: Exec the pods and run the dmesg, you will see output like this:





```
[ 0.000000] Starting gVisor...
[ 0.183366] Creating cloned children...
[ 0.290397] Moving files to filing cabinet...
[ 0.392925] Letting the watchdogs out...
[ 0.452958] Digging up root...
[ 0.937597] Gathering forks...
[ 1.095681] Daemonizing children...
[ 1.306448] Rewriting operating system in Javascript...
[ 1.514936] Reading process obituaries...
[ 1.589958] Waiting for children...
[ 1.892298] Segmenting fault lines...
[ 1.974048] Ready!
```

- A. Mastered  
B. Not Mastered

**Answer:** A

**Explanation:**

Send us your feedback on it.

**NEW QUESTION 13**

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

- A. Mastered  
B. Not Mastered

**Answer:** A

**Explanation:**

Send us your suggestion on it.

**NEW QUESTION 18**

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.

- A. Mastered  
B. Not Mastered

**Answer:** A

**Explanation:**

Send us your Feedback on this.

**NEW QUESTION 20**

.....

## Thank You for Trying Our Product

\* 100% Pass or Money Back

All our products come with a 90-day Money Back Guarantee.

\* One year free update

You can enjoy free update one year. 24x7 online support.

\* Trusted by Millions

We currently serve more than 30,000,000 customers.

\* Shop Securely

All transactions are protected by VeriSign!

**100% Pass Your CKS Exam with Our Prep Materials Via below:**

<https://www.certleader.com/CKS-dumps.html>