

## Exam Questions CKS

Certified Kubernetes Security Specialist (CKS) Exam

<https://www.2passeasy.com/dumps/CKS/>



### 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

Given an existing Pod named nginx-pod running in the namespace test-system, fetch the service-account-name used and put the content in /candidate/KSC00124.txt

Create a new Role named dev-test-role in the namespace test-system, which can perform update operations, on resources of type namespaces.

Create a new RoleBinding named dev-test-role-binding, which binds the newly created Role to the Pod's ServiceAccount ( found in the Nginx pod running in namespace test-system).

- A. Mastered
- B. Not Mastered

**Answer:** A

#### Explanation:

Send us your feedback on it.

### NEW QUESTION 3

Given an existing Pod named test-web-pod running in the namespace test-system

Edit the existing Role bound to the Pod's Service Account named sa-backend to only allow performing get operations on endpoints.

Create a new Role named test-system-role-2 in the namespace test-system, which can perform patch operations, on resources of type statefulsets.

Create a new RoleBinding named test-system-role-2-binding binding the newly created Role to the Pod's ServiceAccount sa-backend.

- A. Mastered
- B. Not Mastered

**Answer:** A

#### Explanation:

Send us your feedback on this.

### NEW QUESTION 4

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 12 days.
- \* 3. at maximum, a number of 8 old audit logs files are retained.
- \* 4. set the maximum size before getting rotated to 200MB

Edit and extend the basic policy to log:

- \* 1. namespaces changes at RequestResponse
  - \* 2. Log the request body of secrets changes in the namespace kube-system.
  - \* 3. Log all other resources in core and extensions at the Request level.
  - \* 4. Log "pods/portforward", "services/proxy" at Metadata level.
  - \* 5. Omit the Stage RequestReceived
- All other requests at the Metadata level

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Kubernetes auditing provides a security-relevant chronological set of records about a cluster. Kube-apiserver performs auditing. Each request on each stage of its execution generates an event, which is then pre-processed according to a certain policy and written to a backend. The policy determines what's recorded and the backends persist the records.

You might want to configure the audit log as part of compliance with the CIS (Center for Internet Security) Kubernetes Benchmark controls.

The audit log can be enabled by default using the following configuration in cluster.yml:

services:

kube-api:

audit\_log:

enabled:true

When the audit log is enabled, you should be able to see the default values at

/etc/kubernetes/audit-policy.yaml

The log backend writes audit events to a file in JSONlines format. You can configure the log audit backend using the following kube-apiserver flags:

- > --audit-log-path specifies the log file path that log backend uses to write audit events. Not specifying thi flag disables log backend. - means standard out
- > --audit-log-maxbackup defines the maximum number of audit log files to retain
- > --audit-log-maxsize defines the maximum size in megabytes of the audit log file before it gets rotated

If your cluster's control plane runs the kube-apiserver as a Pod, remember to mount the location of the policy file and log file, so that audit records are persisted.

For example:-hostPath-to the

--audit-policy-file=/etc/kubernetes/audit-policy.yaml\

--audit-log-path=/var/log/audit.log-

**NEW QUESTION 5**

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 thAe PI server:

- \* a. Ensure the --authorization-mode argument includes RBAC
- \* b. Ensure the --authorization-mode argument includes Node
- \* c. Ensure that the --profiling argumentissettofalse

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 argument is not set to true

Hint: Take the use of Tool Kube-Bench

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

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 - BuildtimeKubernetesapiVersion: 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

--anonymous-auth=false

variable.

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 - BuildtimeKubernetesapiVersion: 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: {}
```

#### NEW QUESTION 6

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 7

Use the kubesecc docker images to scan the given YAML manifest, edit and apply the advised changes, and passed with a score of 4 points.

```
kubesecc-test.yaml
apiVersion: v1
kind: Pod
metadata:
name: kubesecc-demo
spec:
containers:
- name: kubesecc-demo
image: gcr.io/google-samples/node-hello:1.0
securityContext:
readOnlyRootFilesystem:true
Hint: docker run -i kubesecc/kubesecc:512c5e0 scan /dev/stdin< kubesecc-test.yaml
```



- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Send us your feedback on it.

**NEW QUESTION 8**

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.

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

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Send us your feedback on it.

**NEW QUESTION 9**

Create a User named john, create the CSR Request, fetch the certificate of the user after approving it. Create a Role name john-role to list secrets, pods in namespace john

Finally, Create a RoleBinding named john-role-binding to attach the newly created role john-role to the user john in the namespace john.

To Verify: Use the kubectl auth CLI command to verify the permissions.

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

se kubectl to create a CSR and approve it.

Get the list of CSRs:

kubectl get csr

Approve the CSR:

kubectl certificate approve myuser

Get the certificateRetrieve the certificate from the CSR:

kubectl get csr/myuser -o yaml

here are the role and role-binding to give john permission to create NEW\_CRD resource: kubectlapply-froleBindingJohn.yaml--as=john

rolebinding.rbac.authorization.k8s.io/john\_external-rosource-rbcreated

kind:RoleBinding

apiVersion:rbac.authorization.k8s.io/v1

metadata:

name:john\_crd

namespace:development-john

subjects:

-kind:User

name:john

apiGroup:rbac.authorization.k8s.io

roleRef:

kind:ClusterRole

name:crd-creation

kind:ClusterRole

apiVersion:rbac.authorization.k8s.io/v1

metadata:

name:crd-creation

rules:

-apiGroups:["kubernetes-client.io/v1"]

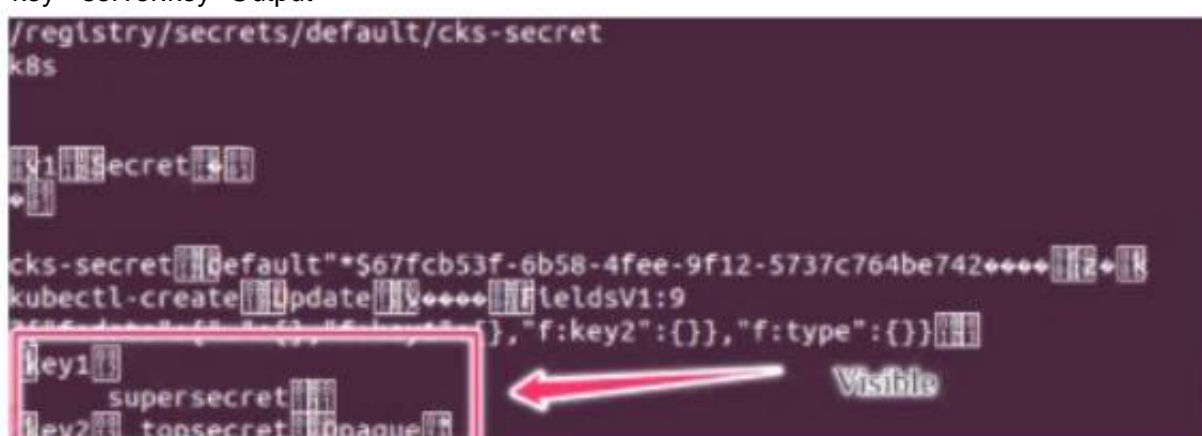
resources:["NEW\_CRD"]

verbs:["create, list, get"]

**NEW QUESTION 10**

Secrets stored in the etcd is not secure at rest, you can use the etcdctl command utility to find the secret value for e.g:ETCDCTL\_API=3 etcdctl get /registry/secrets/default/cks-secret --cacert="ca.crt" --cert="server.crt"

--key="server.key" Output



Using the Encryption Configuration, Create the manifest, which secures the resource secrets using the provider AES-CBC and identity, to encrypt the secret-data at rest and ensure all secrets are encrypted with the new configuration.

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

Send us your feedback on it.

**NEW QUESTION 10**

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 15**

.....

## THANKS FOR TRYING THE DEMO OF OUR PRODUCT

Visit Our Site to Purchase the Full Set of Actual CKS Exam Questions With Answers.

We Also Provide Practice Exam Software That Simulates Real Exam Environment And Has Many Self-Assessment Features. Order the CKS Product From:

<https://www.2passeasy.com/dumps/CKS/>

## Money Back Guarantee

### CKS Practice Exam Features:

- \* CKS Questions and Answers Updated Frequently
- \* CKS Practice Questions Verified by Expert Senior Certified Staff
- \* CKS Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- \* CKS Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year