Linux-Foundation

Exam Questions CKS

Certified Kubernetes Security Specialist (CKS) Exam

About Exambible

Your Partner of IT Exam

Found in 1998

Exambible is a company specialized on providing high quality IT exam practice study materials, especially Cisco CCNA, CCDA, CCNP, CCIE, Checkpoint CCSE, CompTIA A+, Network+ certification practice exams and so on. We guarantee that the candidates will not only pass any IT exam at the first attempt but also get profound understanding about the certificates they have got. There are so many alike companies in this industry, however, Exambible has its unique advantages that other companies could not achieve.

Our Advances

* 99.9% Uptime

All examinations will be up to date.

* 24/7 Quality Support

We will provide service round the clock.

* 100% Pass Rate

Our guarantee that you will pass the exam.

* Unique Gurantee

If you do not pass the exam at the first time, we will not only arrange FULL REFUND for you, but also provide you another exam of your claim, ABSOLUTELY FREE!

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
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 servic 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 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. Mastered
B. Not Mastered

Answer: A

Explanation:
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-<<EOF
apiVersion: v1
kind: Pod
metadata:
name: pause
spec:
containers:
- name: pause
image: k8s.gcr.io/pause
EOF
The output is similar to this:
Error from server (Forbidden): error when creating "STDIN": pods "pause" is forbidden: unable to validate against any pod security policy: []
Create a new ServiceAccount named psp-sa in the namespace default.
$ 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 ClusterRole named prevent-role, which uses the newly created Pod Security Policy prevent-privileged-policy.
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-<<EOF
apiVersion: v1
kind: Pod
metadata:
name: pause
spec:
containers:
- name: pause
image: k8s.gcr.io/pause EOF
The output is similar to this:
Error from server (Forbidden): error when creating "STDIN": pods "pause" is forbidden: unable to validate against any pod security policy: []
Create a new ClusterRoleBinding named prevent-role-binding, which binds the created ClusterRole prevent-role to the created SA psp-sa.
apiVersion:rbac.authorization.k8s.io/v1
This role binding allows "jane" to read pods in the "default" namespace.
You need to already have a Role named "pod-reader" in that namespace.
kind:RoleBinding
metadata:
name:read-pods
namespace:default
subjects:
You can specify more than one "subject"
-kind:User
name:jane# "name" is case sensitive
apiGroup:rbac.authorization.k8s.io
roleRef:
"roleRef" specifies the binding to a Role / ClusterRole
kind:Role#this must be Role or ClusterRole
name:pod-reader# this must match the name of the Role or ClusterRole you wish to bind to
apiGroup:rbac.authorization.k8s.io apiVersion:rbac.authorization.k8s.io/v1
kind:Role
metadata:
namespace:default
name:pod-reader
rules:
-apiGroups:[""]# "" indicates the core API group
resources:["pods"]
verbs:["get","watch","list"]

NEW QUESTION 4
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 5
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.

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Finally, test the configuration by deploying the pod having the image tag as latest. Send us your Feedback on this.

NEW QUESTION 6
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 7
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-getupdate -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.

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Send us your feedback on it.

NEW QUESTION 8
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 10
.....

Relate Links

100% Pass Your CKS Exam with Exambible Prep Materials

https://www.exambible.com/CKS-exam/

Contact us

We are proud of our high-quality customer service, which serves you around the clock 24/7.

Viste - https://www.exambible.com/