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
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
* 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 2
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 3
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 argumentissetasappropriate.
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 argumentisnotsettotrue
Edit the etcd pod specification file $etcdconf on the masternode and either remove the --auto-tls parameter or set it to false.--auto-tls=false
* b. Ensure that the --peer-auto-tls argumentisnotsettotrue
Edit the etcd pod specification file $etcdconf on the masternode and either remove the --peer-auto-tls parameter or set it to false.--peer-auto-tls=false

NEW QUESTION 4
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 5
Use the kubesec docker images to scan the given YAML manifest, edit and apply the advised changes, and passed with a score of 4 points.
kubesec-test.yaml
apiVersion: v1
kind: Pod
metadata:
name: kubesec-demo
spec:
containers:
- name: kubesec-demo
image: gcr.io/google-samples/node-hello:1.0
securityContext:
readOnlyRootFilesystem:true
Hint: docker run -i kubesec/kubesec:512c5e0 scan /dev/stdin< kubesec-test.yaml

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Send us your feedback on it.

NEW QUESTION 6
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 7
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 8
* a. Retrieve the content of the existing secret named default-token-xxxxx in the testing namespace.
Store the value of the token in the token.txt
* b. Create a new secret named test-db-secret in the DB namespace with the following content: username: mysql
password: password@123
Create the Pod name test-db-pod of image nginx in the namespace db that can access test-db-secret via a volume at path /etc/mysql-credentials

A. Mastered
B. Not Mastered

Answer: A

Explanation:
To add a Kubernetes cluster to your project, group, or instance:
Navigate to your:
Project’s Operations > Kubernetes
page, for a project-level cluster.
Group’s Kubernetes
page, for a group-level cluster.
Admin Area > Kubernetes
page, for an instance-level cluster.
Click Add Kubernetes cluster.
Click the Add existing cluster
tab and fill in the details:
Kubernetes cluster name (required) - The name you wish to give the cluster.
Environment scope (required) - The associated environment to this cluster.
API URL (required) - It’s the URL that GitLab uses to access the Kubernetes API. Kubernetes exposes several APIs, we want the “base” URL that is common to all of them. For
example, https://kubernetes.example.com rather than https://kubernetes.example.com/api/v1.
Get the API URL by running this command:
kubectl cluster-info | grep-E'Kubernetes master|Kubernetes control plane'| awk'/http/ {print $NF}'
CA certificate (required) - A valid Kubernetes certificate is needed to authenticate to the cluster.
We use the certificate created by default.
List the secrets with kubectl get secrets, and one should be named similar to default-token-xxxxx. Copy that token name for use below.
Get the certificate by running this command: kubectl get secret <secret name>-ojsonpath="{['data']['ca\.crt']}"

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/