Linux-Foundation

Exam Questions CKS

Certified Kubernetes Security Specialist (CKS) Exam

NEW QUESTION 1
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 2
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 3
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 4
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 5
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 6
Create a network policy named allow-np, that allows pod in the namespace staging to connect to port 80 of other pods in the same namespace.
Ensure that Network Policy:
* 1. Does not allow access to pod not listening on port 80.
* 2. Does not allow access from Pods, not in namespace staging.

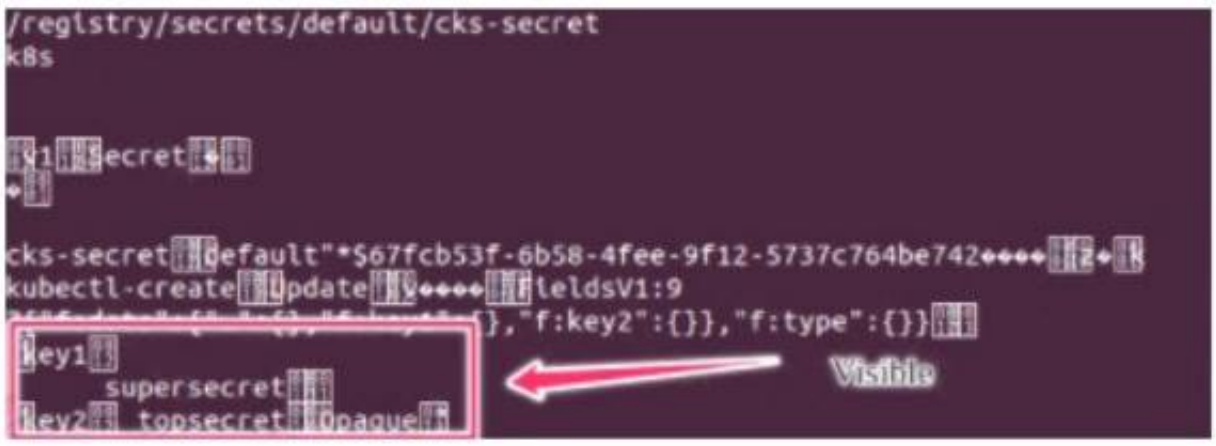
A. Mastered
B. Not Mastered

Answer: A

Explanation:
apiVersion:networking.k8s.io/v1
kind:NetworkPolicy
metadata:
name:network-policy

spec:
podSelector:{} #selects all the pods in the namespace deployed
policyTypes:
-Ingress
ingress:
-ports:#in input traffic allowed only through 80 port only
-protocol:TCP
port:80

NEW QUESTION 7
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 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
.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questons and Answers in PDF Format

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