Linux-Foundation

Exam Questions CKS

Certified Kubernetes Security Specialist (CKS) Exam

NEW QUESTION 1
Create a network policy named restrict-np to restrict to pod nginx-test running in namespace testing. Only allow the following Pods to connect to Pod nginx-test:
* 1. pods in the namespace default
* 2. pods with label version:v1 in any namespace.
Make sure to apply the network policy.

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Send us your Feedback on this.

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
* 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
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 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
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
```

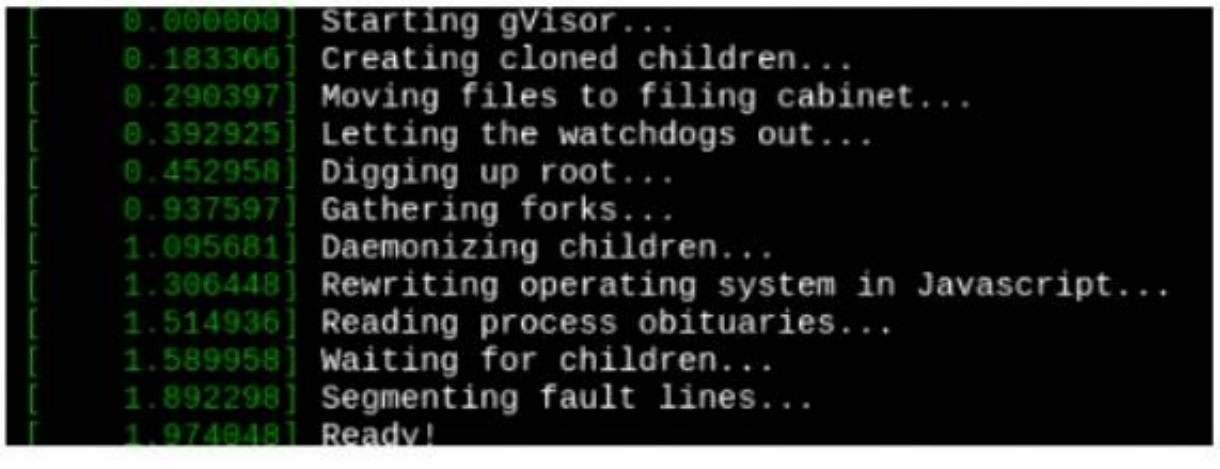
Finally, apply the manifests files and create the Pod specified on it.
Verify: Try to use command ping, top, sh

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Send us your feedback on it.

NEW QUESTION 6
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:



A. Mastered
B. Not Mastered

Answer: A

Explanation:
Send us your feedback on it.

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