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
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
Service is running on port 389 inside the system, find the process-id of the process, and stores the names of all the open-files inside the /candidate/KH77539/files.txt, and also delete the binary.

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Send us your feedback on it.

NEW QUESTION 4
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 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.

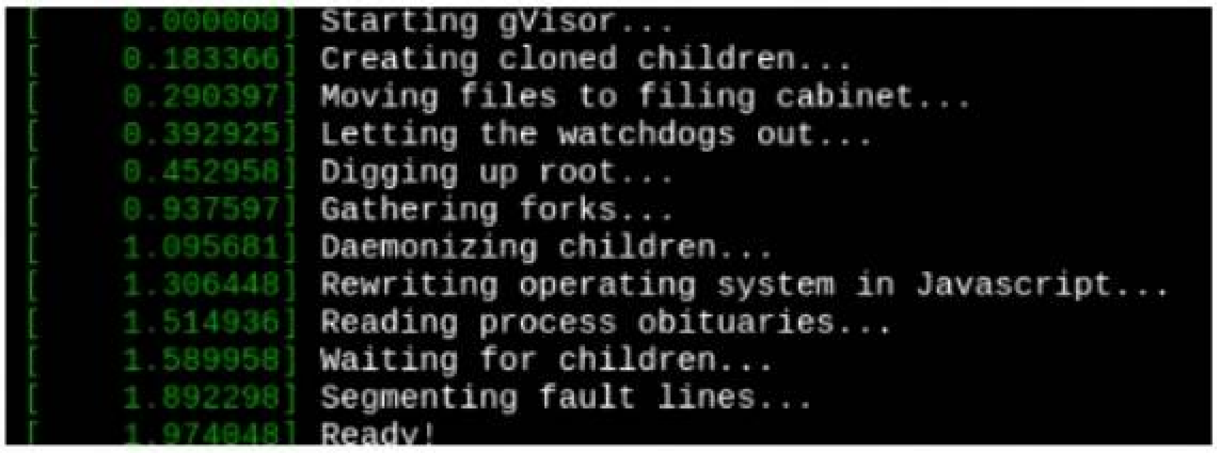
NEW QUESTION 6
A container image scanner is set up on the cluster. Given an incomplete configuration in the directory /etc/kubernetes/confcontrol and a functional container image scanner with HTTPS endpoint https://test-server.local.8081/image_policy
* 1. Enable the admission plugin.
* 2. Validate the 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 7
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:



```
[       0.000000] Starting gVisor...
[       0.183366] Creating cloned children...
[       0.290397] Moving files to filing cabinet...
[       0.392925] Letting the watchdogs out...
[       0.452958] Digging up root...
[       0.937597] Gathering forks...
[       1.095681] Daemonizing children...
[       1.306448] Rewriting operating system in Javascript...
[       1.514936] Reading process obituaries...
[       1.589958] Waiting for children...
[       1.892298] Segmenting fault lines...
[       1.974848] Ready!
```

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Send us your feedback on it.

NEW QUESTION 8
Create a PSP that will only allow the persistentvolumeclaim as the volume type in the namespace restricted.
Create a new PodSecurityPolicy named prevent-volume-policy which prevents the pods which is having different volumes mount apart from persistentvolumeclaim.
Create a new ServiceAccount named psp-sa in the namespace restricted.
Create a new ClusterRole named psp-role, which uses the newly created Pod Security Policy prevent-volume-policy
Create a new ClusterRoleBinding named psp-role-binding, which binds the created ClusterRole psp-role to the created SA psp-sa.
Hint:
Also, Check the Configuration is working or not by trying to Mount a Secret in the pod maifest, it should get failed.
POD Manifest:
* apiVersion: v1
* kind: Pod
* metadata:
* name:
* spec:
* containers:
* - name:
* image:
* volumeMounts:
* - name:
* mountPath:
* volumes:
* - name:
* secret:

* secretName:

A. Mastered
B. Not Mastered

Answer: A

Explanation:

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
name: restricted
annotations:
seccomp.security.alpha.kubernetes.io/allowedProfileNames: 'docker/default,runtime/default'
apparmor.security.beta.kubernetes.io/allowedProfileNames: 'runtime/default' seccomp.security.alpha.kubernetes.io/defaultProfileName: 'runtime/default'
apparmor.security.beta.kubernetes.io/defaultProfileName: 'runtime/default'
spec:
privileged: false
# Required to prevent escalations to root.
allowPrivilegeEscalation: false
# This is redundant with non-root + disallow privilege escalation,
# but we can provide it for defense in depth.
requiredDropCapabilities:
- ALL
# Allow core volume types. volumes:
- 'configMap'
- 'emptyDir'
- 'projected'
- 'secret'
- 'downwardAPI'
# Assume that persistentVolumes set up by the cluster admin are safe to use.
- 'persistentVolumeClaim'
hostNetwork: false
hostIPC: false
hostPID: false
runAsUser:
# Require the container to run without root privileges.
rule: 'MustRunAsNonRoot'
seLinux:
# This policy assumes the nodes are using AppArmor rather than SELinux.
rule: 'RunAsAny'
supplementalGroups:
rule: 'MustRunAs'
ranges:
# Forbid adding the root group.
- min: 1
max: 65535
fsGroup:
rule: 'MustRunAs'
ranges:
# Forbid adding the root group.
- min: 1
max: 65535
readOnlyRootFilesystem: false
```

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/