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
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 thAe PI server:
* a. Ensure the --authorization-mode argument includes RBAC
* b. Ensure the --authorization-mode argument includes Node
* c. Ensure that the --profiling argumentissettofalse
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 argument is not set to true
Hint: Take the use of Tool Kube-Bench

A. Mastered
B. Not Mastered

Answer: A

Explanation:
API server:
Ensure the --authorization-mode argument includes RBAC
Turn on Role Based Access Control.Role Based Access Control (RBAC) allows fine-grained control over the operations that different entities can perform on different objects in the cluster. It is recommended to use the RBAC authorization mode.
Fix - BuildtimeKubernetesapiVersion: v1
kind: Pod
metadata:
creationTimestamp: null
labels:
component: kube-apiserver
tier: control-plane
name: kube-apiserver
namespace: kube-system spec:
containers:
-command:
+ - kube-apiserver
+ - --authorization-mode=RBAC,Node
image: gcr.io/google_containers/kube-apiserver-amd64:v1.6.0
livenessProbe:
failureThreshold:8
httpGet:
host:127.0.0.1
path: /healthz
port:6443
scheme: HTTPS
initialDelaySeconds:15
timeoutSeconds:15
name: kube-apiserver-should-pass
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
Ensure the --authorization-mode argument includes Node
Remediation: Edit the API server pod specification fil/eetc/kubernetes/manifests/kube-apiserver.yaml on
the master node and set the --authorization-mode parameter to a value that includeNs ode.
--authorization-mode=Node,RBAC
Audit:
/bin/ps -ef | grep kube-apiserver | grep -v grep
Expected result:
'Node,RBAC' has 'Node'
Ensure that the --profiling argumentissettofalse
Remediation: Edit the API server pod specification fil/eetc/kubernetes/manifests/kube-apiserver.yaml on the master node and set the below parameter.
--profiling=false
Audit:
/bin/ps -ef | grep kube-apiserver | grep -v grep
Expected result:
'false' is equal to 'false'
Fix all of the following violations that were found against the Kubelet:-
Ensure the --anonymous-auth argumentissettofalse.
Remediation: If using a Kubelet config file, edit the file to set authenticationa:nonymous: enabled to false. If using executable arguments, edit the kubelet service file
/etc/systemd/system/kubelet.service.d/10-kubeadm.conf
on each worker node and set the below parameter
in KUBELET_SYSTEM_PODS_ARGS
--anonymous-auth=false
variable.
Based on your system, restart the kubelet service. For example:
systemctl daemon-reload
systemctl restart kubelet.service
Audit:
/bin/ps -fC kubelet
Audit Config:
/bin/cat /var/lib/kubelet/config.yaml
Expected result:
'false' is equal to 'false'
*2) Ensure that the --authorization-mode argumentissetto Webhook.
Audit
docker inspect kubelet | jq -e'.[0].Args[] | match("--authorization-mode=Webhook").string'
Returned Value: --authorization-mode=Webhook
Fix all of the following violations that were found against the ETCD:
*a. Ensure that the --auto-tls argument is not set to true
Do not use self-signed certificates for TLS. etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should not be available to unauthenticated clients. You should enable the client authentication via valid certificates to secure the access to the etcd service.
Fix - BuildtimeKubernetesapiVersion: v1
kind: Pod
metadata:
annotations:
scheduler.alpha.kubernetes.io/critical-pod:""
creationTimestamp: null
labels:
component: etcd
tier: control-plane
name: etcd
namespace: kube-system
spec:
containers:
-command:
+ - etcd
+ - --auto-tls=true
image: k8s.gcr.io/etcd-amd64:3.2.18
imagePullPolicy: IfNotPresent
livenessProbe:
exec:
command:
- /bin/sh
- -ec
- ETCDCTL_API=3 etcdctl --endpoints=https://[192.168.22.9]:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt
--cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt --key=/etc/kubernetes/pki/etcd/healthcheck-client.key get foo
failureThreshold:8
initialDelaySeconds:15
timeoutSeconds:15
name: etcd-should-fail
resources: {}
volumeMounts:
-mountPath: /var/lib/etcd
name: etcd-data
-mountPath: /etc/kubernetes/pki/etcd
name: etcd-certs
hostNetwork:true
priorityClassName: system-cluster-critical
volumes:
-hostPath:

path: /var/lib/etcd
type: DirectoryOrCreate
name: etcd-data
-hostPath:
path: /etc/kubernetes/pki/etcd
type: DirectoryOrCreate
name: etcd-certs
status: {}

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
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
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 6
Using the runtime detection tool Falco, Analyse the container behavior for at least 20 seconds, using filters that detect newly spawning and executing processes in a single container of Nginx.
store the incident file art /opt/falco-incident.txt, containing the detected incidents. one per line, in the format [timestamp],[uid],[processName]

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Send us your feedback on it.

NEW QUESTION 7
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 8
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 9
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
}
```

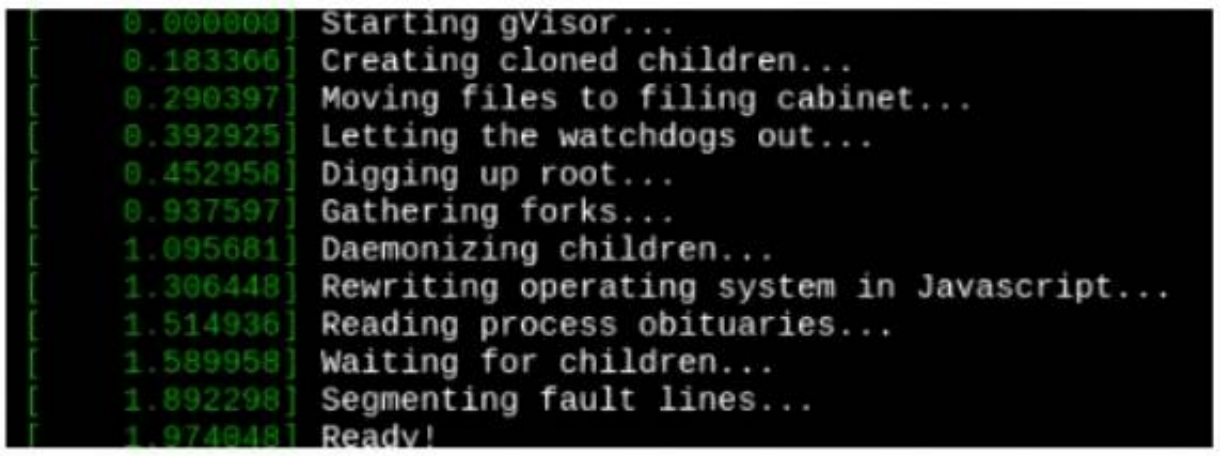
NEW QUESTION 10
Create a Pod name Nginx-pod inside the namespace testing, Create a service for the Nginx-pod named nginx-svc, using the ingress of your choice, run the ingress on tls, secure port.

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Send us your feedback on it.

NEW QUESTION 10
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.

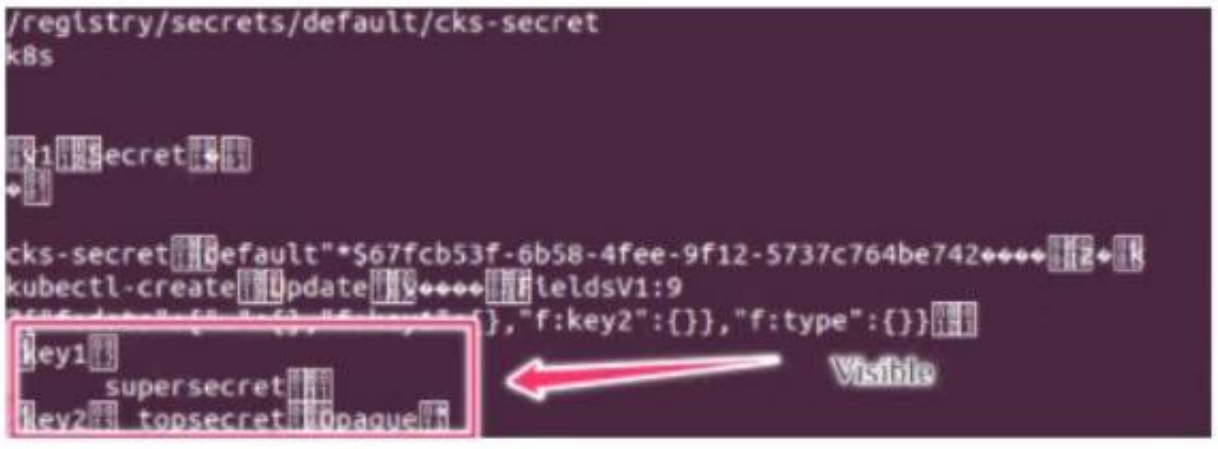
NEW QUESTION 12
Create a User named john, create the CSR Request, fetch the certificate of the user after approving it. Create a Role name john-role to list secrets, pods in namespace john
Finally, Create a RoleBinding named john-role-binding to attach the newly created role john-role to the user john 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 17
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 19
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 23
.....

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