Exam Questions CKS

Certified Kubernetes Security Specialist (CKS) Exam

https://www.2passeasy.com/dumps/CKS/

NEW QUESTION 1
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 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 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 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

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 6
Analyze and edit the given Dockerfile

```
 FROM ubuntu:latest
 RUN apt-getupdate -y
 RUN apt-install nginx -y
 COPY entrypoint.sh /
 ENTRYPOINT ["/entrypoint.sh"]
 USER ROOT
```

Fixing two instructions present in the file being prominent security best practice issues
Analyze and edit the deployment manifest file

```
 apiVersion: v1
 kind: Pod
 metadata:
 name: security-context-demo-2
 spec:
securityContext:
 runAsUser: 1000
 containers:
- name: sec-ctx-demo-2
image: gcr.io/google-samples/node-hello:1.0
 securityContext:
 runAsUser: 0
privileged:True
allowPrivilegeEscalation:false
```

Fixing two fields present in the file being prominent security best practice issues
Don't add or remove configuration settings; only modify the existing configuration settings

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Whenever you need an unprivileged user for any of the tasks, use user test-user with the user id 5487 Send us the Feedback on it.

NEW QUESTION 10
......