Exam Questions CKS

Certified Kubernetes Security Specialist (CKS) Exam

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

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
Enable audit logs in the cluster, To Do so, enable the log backend, and ensure that
* 1. logs are stored at /var/log/kubernetes-logs.txt.
* 2. Log files are retained for 12 days.
* 3. at maximum, a number of 8 old audit logs files are retained.
* 4. set the maximum size before getting rotated to 200MB
Edit and extend the basic policy to log:
* 1. namespaces changes at RequestResponse
* 2. Log the request body of secrets changes in the namespace kube-system.
* 3. Log all other resources in core and extensions at the Request level.
* 4. Log "pods/portforward", "services/proxy" at Metadata level.
* 5. Omit the Stage RequestReceived
All other requests at the Metadata level

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Kubernetes auditing provides a security-relevant chronological set of records about a cluster. Kube-apiserver performs auditing. Each request on each stage of its execution generates an event, which is then pre-processed according to a certain policy and written to a backend. The policy determines what's recorded and the backends persist the records.
You might want to configure the audit log as part of compliance with the CIS (Center for Internet Security) Kubernetes Benchmark controls.
The audit log can be enabled by default using the following configuration in cluster.yml:
services:
kube-api:
audit_log:
enabled:true
When the audit log is enabled, you should be able to see the default values at
/etc/kubernetes/audit-policy.yaml
The log backend writes audit events to a file in JSONlines format. You can configure the log audit backend using the following kube-apiserver flags:
--audit-log-path specifies the log file path that log backend uses to write audit events. Not specifying thi flag disables log backend. - means standard out
--audit-log-maxbackup defines the maximum number of audit log files to retain
--audit-log-maxsize defines the maximum size in megabytes of the audit log file before it gets rotated
If your cluster's control plane runs the kube-apiserver as a Pod, remember to mount the location of the policy file and log file, so that audit records are persisted.
For example:-hostPath-to the
--audit-policy-file=/etc/kubernetes/audit-policy.yaml\
--audit-log-path=/var/log/audit.log-

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 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 4
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 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
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 8
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 9
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
* a. Retrieve the content of the existing secret named default-token-xxxxx in the testing namespace.
Store the value of the token in the token.txt
* b. Create a new secret named test-db-secret in the DB namespace with the following content: username: mysql
password: password@123
Create the Pod name test-db-pod of image nginx in the namespace db that can access test-db-secret via a volume at path /etc/mysql-credentials

A. Mastered
B. Not Mastered

Answer: A

Explanation:
To add a Kubernetes cluster to your project, group, or instance:
Navigate to your:
Project’s Operations > Kubernetes
page, for a project-level cluster.
Group’s Kubernetes
page, for a group-level cluster.
Admin Area > Kubernetes
page, for an instance-level cluster.
Click Add Kubernetes cluster.
Click the Add existing cluster
tab and fill in the details:
Kubernetes cluster name (required) - The name you wish to give the cluster.
Environment scope (required) - The associated environment to this cluster.
API URL (required) - It’s the URL that GitLab uses to access the Kubernetes API. Kubernetes exposes several APIs, we want the “base” URL that is common to all of them. For
example, https://kubernetes.example.com rather than https://kubernetes.example.com/api/v1.
Get the API URL by running this command:
kubectl cluster-info | grep-E'Kubernetes master|Kubernetes control plane'| awk'/http/ {print $NF}'

CA certificate (required) - A valid Kubernetes certificate is needed to authenticate to the cluster.
We use the certificate created by default.
List the secrets with kubectl get secrets, and one should be named similar to default-token-xxxxx. Copy that token name for use below.
Get the certificate by running this command: kubectl get secret <secret name>-ojsonpath="{['data']['ca\.crt']}"

NEW QUESTION 10
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 14
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 18
use the Trivy to scan the following images,
* 1. amazonlinux:1
* 2. k8s.gcr.io/kube-controller-manager:v1.18.6
Look for images with HIGH or CRITICAL severity vulnerabilities and store the output of the same in
/opt/trivy-vulnerable.txt

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Send us your suggestion on it.

NEW QUESTION 23
Before Making any changes build the Dockerfile with tag base:v1 Now Analyze and edit the given Dockerfile(based on ubuntu 16:04)
Fixing two instructions present in the file, Check from Security Aspect and Reduce Size point of view.
Dockerfile:
```
 FROM ubuntu:latest
 RUN apt-getupdate -y
 RUN apt install nginx -y
 COPY entrypoint.sh /
 RUN useradd ubuntu
 ENTRYPOINT ["/entrypoint.sh"]
 USER ubuntu
```
entrypoint.sh
```
 #!/bin/bash
```

echo"Hello from CKS"
After fixing the Dockerfile, build the docker-image with the tag base:v2 To Verify: Check the size of the image before and after the build.

A. Mastered
B. Not Mastered

Answer: A

Explanation:
Send us your feedback on it.

NEW QUESTION 27
......