

CKS Dumps

Certified Kubernetes Security Specialist (CKS) Exam

<https://www.certleader.com/CKS-dumps.html>



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

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

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 at /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 5

* 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 6

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 7

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 manifest, 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 8

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 10

.....

Thank You for Trying Our Product

* 100% Pass or Money Back

All our products come with a 90-day Money Back Guarantee.

* One year free update

You can enjoy free update one year. 24x7 online support.

* Trusted by Millions

We currently serve more than 30,000,000 customers.

* Shop Securely

All transactions are protected by VeriSign!

100% Pass Your CKS Exam with Our Prep Materials Via below:

<https://www.certleader.com/CKS-dumps.html>