

CKS Dumps

Certified Kubernetes Security Specialist (CKS) Exam

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



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

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

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]*}/** 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[fs]*/** 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 5

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 at /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 6

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 7

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 8

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 9

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 10

On the Cluster worker node, enforce the prepared AppArmor profile

```
#include<tunables/global>
profile nginx-deny flags=(attach_disconnected) {
```

```
#include<abstractions/base>
file,
# Deny all file writes.
deny/** w,
}
EOF'
```

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 make a file inside the directory which is restricted.

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Send us your Feedback on this.

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>