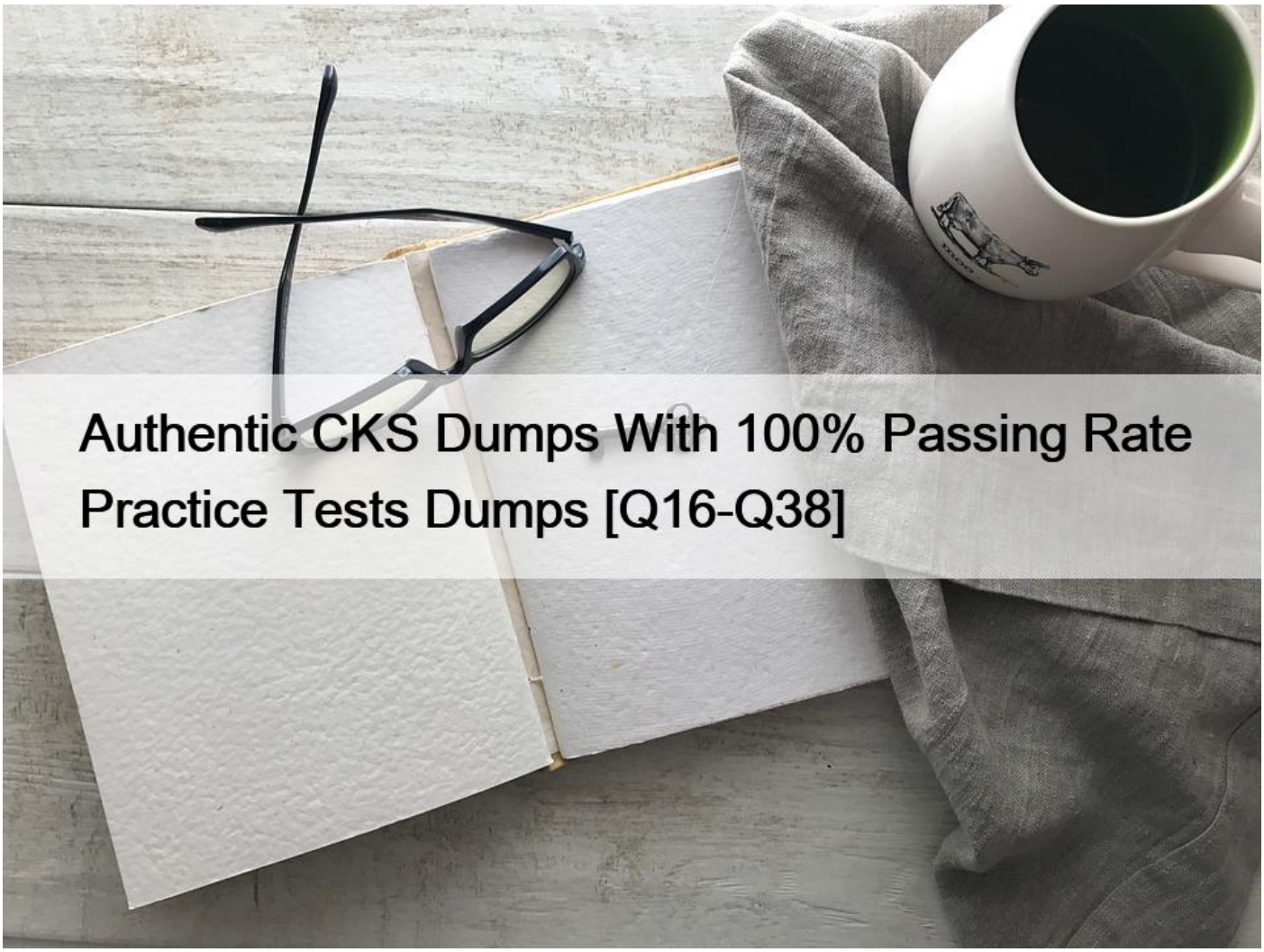
Authentic CKS Dumps With 100% Passing Rate Practice Tests Dumps [Q16-Q38



Authentic CKS Dumps With 100% Passing Rate Practice Tests Dumps
Linux Foundation CKS Real Exam Questions Guaranteed Updated Dump from BraindumpsIT

Q16. SIMULATION

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 retained for 5 days.

3. at maximum, a number of 10 old audit logs files are retained.

Edit and extend the basic policy to log:

1. Cronjobs changes at RequestResponse

2. Log the request body of deployments changes in the namespace kube-system.

3. Log all other resources in core and extensions at the Request level.

4. Don’t log watch requests by the “system:kube-proxy” on endpoints or
* Send us the Feedback on it.

Q17. 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]

Q18. SIMULATION

Given an existing Pod named nginx-pod running in the namespace test-system, fetch the service-account-name used and put the content in /candidate/KSC00124.txt Create a new Role named dev-test-role in the namespace test-system, which can perform update operations, on resources of type namespaces.

Create a new RoleBinding named dev-test-role-binding, which binds the newly created Role to the Pod’s ServiceAccount (found in the Nginx pod running in namespace test-system).
* Sendusyourfeedbackonit

Q19. 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.

Q20. SIMULATION

Service is running on port 389 inside the system, find the process-id of the process, and stores the names of all the open-files inside the /candidate/KH77539/files.txt, and also delete the binary.
* Send us your feedback on it.

Q21. SIMULATION

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
* Send us the Feedback on it.

Q22. 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 the –authorization-mode argument includes RBAC b. Ensure the –authorization-mode argument includes Node c. Ensure that the –profiling argument is set to false Fix all of the following violations that were found against the Kubelet:- a. Ensure the –anonymous-auth argument is set to false.

b. Ensure that the –authorization-mode argument is set to 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
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 – Buildtime

Kubernetes

apiVersion: 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 file /etc/kubernetes/manifests/kube-apiserver.yaml on the master node and set the –authorization-mode parameter to a value that includes Node.

–authorization-mode=Node,RBAC

Audit:

/bin/ps -ef | grep kube-apiserver | grep -v grep

Expected result:

‘Node,RBAC’ has ‘Node’

Ensure that the –profiling argument is set to false

Remediation: Edit the API server pod specification file /etc/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 argument is set to false.

Remediation: If using a Kubelet config file, edit the file to set authentication: anonymous: 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 variable.

–anonymous-auth=false

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 argument is set to 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 – Buildtime

Kubernetes

apiVersion: 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: {}

Q23. 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.
* Send us your Feedback on this.

Q24. SIMULATION

Analyze and edit the given Dockerfile

FROM ubuntu:latest

RUN apt-get update -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 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.

Q25. You can switch the cluster/configuration context using the following command:

[desk@cli] $ kubectl config use-context prod-account

Context:

A Role bound to a Pod’s ServiceAccount grants overly permissive permissions. Complete the following tasks to reduce the set of permissions.

Task:

Given an existing Pod named web-pod running in the namespace database.

1. Edit the existing Role bound to the Pod’s ServiceAccount test-sa to only allow performing get operations, only on resources of type Pods.

2. Create a new Role named test-role-2 in the namespace database, which only allows performing update operations, only on resources of type statuefulsets.

3. Create a new RoleBinding named test-role-2-bind binding the newly created Role to the Pod’s ServiceAccount.

Note: Don’t delete the existing RoleBinding.

$ k edit role test-role -n database

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

creationTimestamp: “2021-06-04T11:12:23Z”

name: test-role

namespace: database

resourceVersion: “1139”

selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/database/roles/test-role uid: 49949265-6e01-499c-94ac-5011d6f6a353 rules:

– apiGroups:

– “”

resources:

– pods

verbs:

– * # Delete

– get # Fixed

$ k create role test-role-2 -n database –resource statefulset –verb update

$ k create rolebinding test-role-2-bind -n database –role test-role-2 –serviceaccount=database:test-sa Explanation

[desk@cli]$ k get pods -n database

NAME READY STATUS RESTARTS AGE LABELS

web-pod 1/1 Running 0 34s run=web-pod

[desk@cli]$ k get roles -n database

test-role

[desk@cli]$ k edit role test-role -n database

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

creationTimestamp: “2021-06-13T11:12:23Z”

name: test-role

namespace: database

resourceVersion: “1139”

selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/database/roles/test-role uid: 49949265-6e01-499c-94ac-5011d6f6a353 rules:

– apiGroups:

– “”

resources:

– pods

verbs:

– “*” # Delete this

– get # Replace by this

[desk@cli]$ k create role test-role-2 -n database –resource statefulset –verb update role.rbac.authorization.k8s.io/test-role-2 created [desk@cli]$ k create rolebinding test-role-2-bind -n database –role test-role-2 –serviceaccount=database:test-sa rolebinding.rbac.authorization.k8s.io/test-role-2-bind created Reference: https://kubernetes.io/docs/reference/access-authn-authz/rbac/ role.rbac.authorization.k8s.io/test-role-2 created

[desk@cli]$ k create rolebinding test-role-2-bind -n database –role test-role-2 –serviceaccount=database:test-sa rolebinding.rbac.authorization.k8s.io/test-role-2-bind created

[desk@cli]$ k create role test-role-2 -n database –resource statefulset –verb update role.rbac.authorization.k8s.io/test-role-2 created [desk@cli]$ k create rolebinding test-role-2-bind -n database –role test-role-2 –serviceaccount=database:test-sa rolebinding.rbac.authorization.k8s.io/test-role-2-bind created Reference: https://kubernetes.io/docs/reference/access-authn-authz/rbac/

Q26. 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-get update -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.

Q27. You can switch the cluster/configuration context using the following command:

[desk@cli] $ kubectl config use-context dev

A default-deny NetworkPolicy avoid to accidentally expose a Pod in a namespace that doesn’t have any other NetworkPolicy defined.

Task: Create a new default-deny NetworkPolicy named deny-network in the namespace test for all traffic of type Ingress + Egress The new NetworkPolicy must deny all Ingress + Egress traffic in the namespace test.

Apply the newly created default-deny NetworkPolicy to all Pods running in namespace test.

You can find a skeleton manifests file at /home/cert_masters/network-policy.yaml
master1 $ k get pods -n test –show-labels

NAME READY STATUS RESTARTS AGE LABELS

test-pod 1/1 Running 0 34s role=test,run=test-pod

testing 1/1 Running 0 17d run=testing

$ vim netpol.yaml

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: deny-network

namespace: test

spec:

podSelector: {}

policyTypes:

– Ingress

– Egress

master1 $ k apply -f netpol.yaml

Explanation

controlplane $ k get pods -n test –show-labels

NAME READY STATUS RESTARTS AGE LABELS

test-pod 1/1 Running 0 34s role=test,run=test-pod

testing 1/1 Running 0 17d run=testing

master1 $ vim netpol1.yaml

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: deny-network

namespace: test

spec:

podSelector: {}

policyTypes:

– Ingress

– Egress

master1 $ k apply -f netpol1.yaml Reference: https://kubernetes.io/docs/concepts/services-networking/network-policies/ Reference:

master1 $ k apply -f netpol1.yaml Reference: https://kubernetes.io/docs/concepts/services-networking/network-policies/ Explanation controlplane $ k get pods -n test –show-labels NAME READY STATUS RESTARTS AGE LABELS test-pod 1/1 Running 0 34s role=test,run=test-pod testing 1/1 Running 0 17d run=testing master1 $ vim netpol1.yaml apiVersion: networking.k8s.io/v1 kind: NetworkPolicy metadata:

name: deny-network

namespace: test

spec:

podSelector: {}

policyTypes:

– Ingress

– Egress

master1 $ k apply -f netpol1.yaml Reference: https://kubernetes.io/docs/concepts/services-networking/network-policies/ master1 $ k apply -f netpol1.yaml Reference: https://kubernetes.io/docs/concepts/services-networking/network-policies/

Q28. SIMULATION

A container image scanner is set up on the cluster.

Given an incomplete configuration in the directory

/etc/Kubernetes/confcontrol and a functional container image scanner with HTTPS endpoint https://acme.local.8081/image_policy

1. Enable the admission plugin.

2. Validate the control configuration and change it to implicit deny.

Finally, test the configuration by deploying the pod having the image tag as the latest.
* Send us the Feedback on it.

Q29. You can switch the cluster/configuration context using the following command:

[desk@cli] $ kubectl config use-context test-account

Task: 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 5 days

3. at maximum, a number of 10 old audit log files are retained

A basic policy is provided at /etc/Kubernetes/logpolicy/audit-policy.yaml. It only specifies what not to log.

Note: The base policy is located on the cluster’s master node.

Edit and extend the basic policy to log:

1. Nodes changes at RequestResponse level

2. The request body of persistentvolumes changes in the namespace frontend

3. ConfigMap and Secret changes in all namespaces at the Metadata level Also, add a catch-all rule to log all other requests at the Metadata level Note: Don’t forget to apply the modified policy.
$ vim /etc/kubernetes/log-policy/audit-policy.yaml

– level: RequestResponse

userGroups: [“system:nodes”]

– level: Request

resources:

– group: “” # core API group

resources: [“persistentvolumes”]

namespaces: [“frontend”]

– level: Metadata

resources:

– group: “”

resources: [“configmaps”, “secrets”]

– level: Metadata

$ vim /etc/kubernetes/manifests/kube-apiserver.yaml

Add these

– –audit-policy-file=/etc/kubernetes/log-policy/audit-policy.yaml

– –audit-log-path=/var/log/kubernetes/logs.txt

– –audit-log-maxage=5

– –audit-log-maxbackup=10

Explanation

[desk@cli] $ ssh master1

[master1@cli] $ vim /etc/kubernetes/log-policy/audit-policy.yaml

apiVersion: audit.k8s.io/v1 # This is required.

kind: Policy

Don’t generate audit events for all requests in RequestReceived stage.

omitStages:

– “RequestReceived”

rules:

Don’t log watch requests by the “system:kube-proxy” on endpoints or services

– level: None

users: [“system:kube-proxy”]

verbs: [“watch”]

resources:

– group: “” # core API group

resources: [“endpoints”, “services”]

Don’t log authenticated requests to certain non-resource URL paths.

– level: None

userGroups: [“system:authenticated”]

nonResourceURLs:

– “/api*” # Wildcard matching.

– “/version”

Add your changes below

– level: RequestResponse

userGroups: [“system:nodes”] # Block for nodes

– level: Request

resources:

– group: “” # core API group

resources: [“persistentvolumes”] # Block for persistentvolumes

namespaces: [“frontend”] # Block for persistentvolumes of frontend ns

– level: Metadata

resources:

– group: “” # core API group

resources: [“configmaps”, “secrets”] # Block for configmaps & secrets

– level: Metadata # Block for everything else

[master1@cli] $ vim /etc/kubernetes/manifests/kube-apiserver.yaml

apiVersion: v1

kind: Pod

metadata:

annotations:

kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 10.0.0.5:6443 labels:

component: kube-apiserver

tier: control-plane

name: kube-apiserver

namespace: kube-system

spec:

containers:

– command:

– kube-apiserver

– –advertise-address=10.0.0.5

– –allow-privileged=true

– –authorization-mode=Node,RBAC

– –audit-policy-file=/etc/kubernetes/log-policy/audit-policy.yaml #Add this

– –audit-log-path=/var/log/kubernetes/logs.txt #Add this

– –audit-log-maxage=5 #Add this

– –audit-log-maxbackup=10 #Add this

…

output truncated

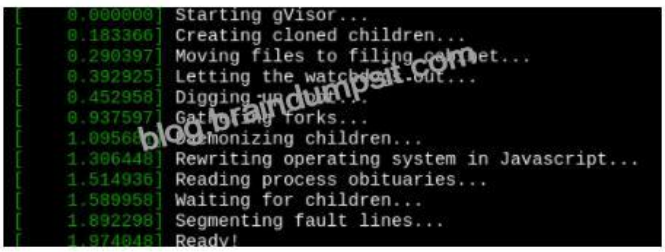
Note: log volume & policy volume is already mounted in vim /etc/kubernetes/manifests/kube-apiserver.yaml so no need to mount it. Reference: https://kubernetes.io/docs/tasks/debug-application-cluster/audit/ Note: log volume & policy volume is already mounted in vim /etc/kubernetes/manifests/kube-apiserver.yaml so no need to mount it. Reference: https://kubernetes.io/docs/tasks/debug-application-cluster/audit/

Q30. SIMULATION

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:-



* Send us your feedback on it.

Q31. Given an existing Pod named nginx-pod running in the namespace test-system, fetch the service-account-name used and put the content in /candidate/KSC00124.txt Create a new Role named dev-test-role in the namespace test-system, which can perform update operations, on resources of type namespaces.
* Create a new RoleBinding named dev-test-role-binding, which binds the newly created Role to the Pod’s ServiceAccount (found in the Nginx pod running in namespace test-system).

Q32. SIMULATION

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-get update -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.
* Send us the Feedback on it.

Q33. Given an existing Pod named test-web-pod running in the namespace test-system Edit the existing Role bound to the Pod’s Service Account named sa-backend to only allow performing get operations on endpoints.

Create a new Role named test-system-role-2 in the namespace test-system, which can perform patch operations, on resources of type statefulsets.
* Create a new RoleBinding named test-system-role-2-binding binding the newly created Role to the Pod’s ServiceAccount sa-backend.

Q34. SIMULATION

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
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 this flag disables log backend. – means standard out

–audit-log-maxage defined the maximum number of days to retain old audit log files

–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 hostPath to the location of the policy file and log file, so that audit records are persisted. For example:

–audit-policy-file=/etc/kubernetes/audit-policy.yaml

–audit-log-path=/var/log/audit.log

Q35. A container image scanner is set up on the cluster.

Given an incomplete configuration in the directory

/etc/Kubernetes/confcontrol and a functional container image scanner with HTTPS endpoint https://acme.local.8081/image_policy
* 1. Enable the admission plugin.
2. Validate the control configuration and change it to implicit deny.

Finally, test the configuration by deploying the pod having the image tag as the latest.

Q36. 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.

Q37. 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:
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

Q38. SIMULATION

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 argument is set to true.

b. Ensure that the admission control plugin PodSecurityPolicy is set.

c. Ensure that the –kubelet-certificate-authority argument is set as appropriate.

Fix all of the following violations that were found against the Kubelet:- a. Ensure the –anonymous-auth argument is set to false.

b. Ensure that the –authorization-mode argument is set to 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

b. Ensure that the –peer-auto-tls argument is not set to true

Hint: Take the use of Tool Kube-Bench
Fix all of the following violations that were found against the API server:- a. Ensure that the RotateKubeletServerCertificate argument is set to true.

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 PodSecurityPolicy is set.

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 argument is set as appropriate.

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 argument is not set to true

Edit the etcd pod specification file $etcdconf on the master node and either remove the –auto-tls parameter or set it to false. –auto-tls=false b. Ensure that the –peer-auto-tls argument is not set to true Edit the etcd pod specification file $etcdconf on the master node and either remove the –peer-auto-tls parameter or set it to false. –peer-auto-tls=false

Verified Pass CKS Exam in First Attempt Guaranteed: https://www.braindumpsit.com/CKS_real-exam.html]