Linux Foundation

CKS

Certified Kubernetes Security Specialist (CKS)

Version: 4.0

[Total Questions: 44]

Web: www.marks4sure.com

Email: support@marks4sure.com

IMPORTANT NOTICE

Feedback

We have developed quality product and state-of-art service to ensure our customers interest. If you have any suggestions, please feel free to contact us at feedback@marks4sure.com

Support

If you have any questions about our product, please provide the following items:

- exam code
- screenshot of the question
- login id/email

please contact us at support@marks4sure.com and our technical experts will provide support within 24 hours.

Copyright

Question #:1

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][^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[^s]*/** 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

Send us your feedback on it.

Question #:2

Fix all issues viaconfiguration 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 argumentissettofalse

Fix all of the following violations that were found against the **Kubelet**:-

- a. Ensure the --anonymous-auth argumentissettofalse.
- b. Ensure thatthe --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

See theExplanation below.

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 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 argumentissettofalse

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 argumentissettofalse.

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 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 useself-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: {}

Question #:3

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 Rolenamed 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.

Send us your feedback on this.

Question #:4

Cluster: dev

Master node: **master1**

Worker node: **worker1**

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

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

Task:

Retrieve the content of the existing secret named **adam** in the **safe** namespace.

Store the username field in a file names **/home/cert-masters/username.txt**, and the password field in a file named **/home/cert-masters/password.txt**.

1. You must create both files; they don't exist yet.

2. Do not use/modify the created files in the following steps, create new temporary files if needed.

Create a new secret names **newsecret** in the **safe** namespace, with the following content:

Username: **dbadmin**

Password: **moresecurepas**

Finally, create a new Pod that has access to the secret **newsecret** via a volume:

- Namespace:safe
- Pod name:mysecret-pod
- Container name:db-container
- Image:redis
- Volume name:secret-vol
- Mount path:/etc/mysecret

See the explanation below

Explanation

1. Get the secret, decrypt it & save in filesk get secret adam -n safe -o yaml

2. Create new secret using --from-literal[desk@cli] $k create secret generic newsecret -n safe --from-literal=username=dbadmin --from-literal=password=moresecurepass

3. Mount it as volume of db-container of mysecret-pod

Explanation[desk@cli] $k get secret adam -n safe -o yamlText Description automatically generated[desk@cli] $ echo "Y2VydC1tYXN0ZXJz" | base64 -d | tee -a /home/cert-masters/username.txt

```
controlplane $ k get secret adam -n safe -o yaml
apiVersion: v1
data:
  password: c2VjcmV0cGFzcw==
  username: Y2VydC1tYXN0ZXJz
kind: Secret
metadata:
  creationTimestamp: "2021-06-13T11:56:32Z"
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:data:
        .: {}
        f:password: {}
        f:username: {}
      f:type: {}
    manager: kubectl-create
    operation: Update
    time: "2021-06-13T11:56:32Z"
  name: adam
  namespace: safe
  resourceVersion: "6405"
  selfLink: /api/v1/namespaces/safe/secrets/adam
  uid: da04fa1e-22a5-49d0-95d5-d73e40542033
type: Opaque
```

```
controlplane $ echo "Y2VydC1tYXN0ZXJz" | base64 -d | tee -a /home/cert-masters/username.txt
cert-masterscontrolplane $
```

[desk@cli] $echo "c2VjcmV0cGFzcw==" | base64 -d | tee -a /home/cert-masters/password.txt

```
controlplane $ echo "c2VjcmV0cGFzcw==" | base64 -d | tee -a /home/cert-masters/password.txt
secretpasscontrolplane $
```

[desk@cli] $k create secret generic newsecret -n safe --from-literal=username=dbadmin --from-literal=password=moresecurepasssecret/newsecret created[desk@cli] $vim /home/certs_masters/secret-pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: mysecret-pod

namespace: safe

labels:

run: mysecret-pod

spec:

containers:

- name: db-container

image: redis

volumeMounts:

- name: secret-vol

mountPath: /etc/mysecret

readOnly: true

volumes:

- name: secret-vol

secret:

secretName: newsecret

[desk@cli] $ k apply -f /home/certs_masters/secret-pod.yamlpod/mysecret-pod created[desk@cli] $ k exec -it mysecret-pod -n safe – cat /etc/mysecret/usernamedbadmin

```
controlplane $ k exec -it mysecret-pod -n safe -- cat /etc/mysecret/username
dbadmincontrolplane $
```

[desk@cli] $ k exec -it mysecret-pod -n safe – cat /etc/mysecret/passwordmoresecurepas

```
controlplane $ k exec -it mysecret-pod -n safe -- cat /etc/mysecret/password
moresecurepasscontrolplane $
```

Question #:5

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.

See the explanation below:

Explanation

apiVersion:networking.k8s.io/v1

kind:NetworkPolicy

metadata:

name:network-policy

spec:

podSelector:{} #selects all the pods in thenamespace deployed

policyTypes:

-Ingress

ingress:

-ports:#in input traffic allowed only through 80 port only

-protocol:TCP

port:80

Question #:6

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 your feedback on it.

Question #:7

Cluster: **admission-cluster**

Master node: **master**

Worker node: **worker1**

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

[desk@cli] $ kubectl config use-context admission-cluster

Context:

A container image scanner is set up on the cluster, but it's not yet fully integrated into the cluster's configuration. When complete, the container image scanner shall scan for and reject the use of vulnerable images.

Task:

You have to complete the entire task on the cluster's master node, where all services and files have been prepared and placed.

Given an incomplete configuration in directory /etc/Kubernetes/config and a functional container image scanner with HTTPS endpoint https://imagescanner.local:8181/image_policy:

1. Enable the necessary plugins to create an image policy

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

3. Edit the configuration to point to the provided HTTPS endpoint correctly

Finally, test if the configuration is working by trying to deploy the vulnerable resource /home/cert_masters/test-pod.yml

Note: You can find the container image scanner's log file at /var/log/policy/scanner.log

See the explanation below

Explanation

[master@cli] $ cd /etc/Kubernetes/config1. Edit kubeconfig to explicity deny[master@cli] $ vim kubeconfig.json"defaultAllow": false # Change to false2. fix server parameter by taking its value from ~/.kube/config[master@cli] $cat /etc/kubernetes/config/kubeconfig.yaml | grep serverserver:3. Enable **ImagePolicyWebhook**[master@cli] $ vim /etc/kubernetes/manifests/kube-apiserver.yaml-
--enable-admission-plugins=NodeRestriction,**ImagePolicyWebhook** # Add this-
--admission-control-config-file=/etc/kubernetes/config/kubeconfig.json # Add this

Explanation[desk@cli] $ ssh master[master@cli] $ cd /etc/Kubernetes/config[master@cli] $ vim kubeconfig.json

{

"imagePolicy": {

"kubeConfigFile": "/etc/kubernetes/config/kubeconfig.yaml",

"allowTTL": 50,

"denyTTL": 50,

"retryBackoff": 500,

"defaultAllow": true # Delete this

"defaultAllow": false # Add this

}

}

Text Description automatically generated[master@cli] $cat /etc/kubernetes/config/kubeconfig.yamlText Description automatically generated

```
{
  "imagePolicy": {
    "kubeConfigFile": "/etc/kubernetes/config/kubeconfig.yaml",
    "allowTTL": 50,
    "denyTTL": 50,
    "retryBackoff": 500,
    "defaultAllow": true    # Delete this
    "defaultAllow": false   # Add this
  }
}
```

```
apiVersion: v1
kind: Config
clusters:
  - cluster:
      certificate-authority: /etc/kubernetes/config/ca.pem
      server:
    name: kubernetes
  - cluster:
contexts:
- context:
    cluster: kubernetes
    user: kube-admin
  name: webhook
current-context: webhook
users:
  - name: kube-admin
    user:
      client-certificate: /etc/kubernetes/config/cert.pem
      client-key: /etc/kubernetes/config/key.pem
```

Note: We can see a missing value here, so how from where i can get this value[master@cli] $cat ~/.kube/config | grep serveror[master@cli] $cat /etc/kubernetes/manifests/kube-apiserver.yaml

```
controlplane $ cat ~/.kube/config | grep server
    server: https://172.17.0.36:6443
```

[master@cli] $vim /etc/kubernetes/config/kubeconfig.yaml

Text Description automatically generated

```
apiVersion: v1
kind: Config
clusters:
  - cluster:
      certificate-authority: /etc/kubernetes/config/ca.pem
      server: https://172.17.0.36:6443    #Add this
    name: kubernetes
  - cluster:
contexts:
- context:
    cluster: kubernetes
    user: kube-admin
  name: webhook
current-context: webhook
users:
  - name: kube-admin
    user:
      client-certificate: /etc/kubernetes/config/cert.pem
      client-key: /etc/kubernetes/config/key.pem
```

Question #:8

Create a PSP that will only allow thepersistentvolumeclaim 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 tothe 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:

See the Explanation below:

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,
# butwe 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 thecluster 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 policyassumes 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

Question #:9

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 theoutput of the same in /opt/trivy-vulnerable.txt

Send us your suggestion on it.

Question #:10

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.

See the explanation below

Explanation

$ 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.yamlAdd 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

Question #:11

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.

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.

Question #:12

Create a new NetworkPolicy named deny-all in the namespace testing which denies all traffic of type ingress and egress traffic

See the explanation below:

Explanation

You can create a "default" isolation policyfor 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 traffic You can create a "default" policy for a namespace which

prevents all ingress AND egress traffic bycreating 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.

Question #:13

Context:

Cluster: **gvisor**

Master node: **master1**

Worker node: **worker1**

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

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

Context: This cluster has been prepared to support runtime handler, runsc as well as traditional one.

Task:

Create a RuntimeClass named **not-trusted** using the prepared runtime handler names **runsc**.

Update all Pods in the namespace server to run on **newruntime**.

See the explanation below

Explanation

Find all the pods/deployment and edit runtimeClassName parameter to not-trusted under spec[desk@cli] $ k edit deploy nginx

- spec:
  - runtimeClassName:not-trusted.# Add this

Explanation[desk@cli] $vim runtime.yaml

apiVersion: node.k8s.io/v1

kind: RuntimeClass

metadata:

name: not-trusted

handler: runsc

[desk@cli] $ k apply -f runtime.yaml[desk@cli] $ k get pods

NAME READY STATUS RESTARTS AGE

nginx-6798fc88e8-chp6r 1/1 Running 0 11m

nginx-6798fc88e8-fs53n 1/1 Running 0 11m

nginx-6798fc88e8-ndved 1/1 Running 0 11m

[desk@cli] $ k get deploy

NAME READY UP-TO-DATE AVAILABLE AGE

nginx 3/3 11 3 5m

[desk@cli] $ k edit deploy nginx

Text Description automatically generated

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  strategy: {}
  template:
    metadata:
      labels:
        app: nginx
    spec:
      runtimeClassName: not-trusted    # Add this
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
```

Question #:14

Context:

Cluster: **prod**

Master node: **master1**

Worker node: **worker1**

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

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

Task:

Analyse and edit the given Dockerfile (based on the **ubuntu:18:04** image)

/home/cert_masters/Dockerfile fixing two instructions present in the file being prominent security/best-practice issues.

Analyse and edit the given manifest file

/home/cert_masters/mydeployment.yaml fixing two fields present in the file being prominent security/best-practice issues.

Note: Don't add or remove configuration settings; only modify the existing configuration settings, so that two configuration settings each are no longer security/best-practice concerns.

Should you need an unprivileged user for any of the tasks, use user **nobody** with user id **65535**

See the explanation below

Explanation

1. For Dockerfile: Fix the image version & user name in Dockerfile2. For mydeployment.yaml : Fix security contexts

Explanation[desk@cli] $ vim /home/cert_masters/Dockerfile

FROM ubuntu:latest # Remove this

FROM ubuntu:18.04 # Add this

USER root # Remove this

USER nobody # Add this

RUN apt get install -y lsof=4.72 wget=1.17.1 nginx=4.2

ENV ENVIRONMENT=testing

USER root # Remove this

USER nobody # Add this

CMD ["nginx -d"]

Text Description automatically generated

```
FROM ubuntu:latest  # Remove this
FROM ubuntu:18.04   # Add this
USER root           # Remove this
USER nobody         # Add this
RUN apt get install -y lsof=4.72 wget=1.17.1 nginx=4.2
ENV  ENVIRONMENT=testing
USER root           # Remove this
USER nobody         # Add this
CMD ["nginx -d"]
```

[desk@cli] $ vim **/home/cert_masters/mydeployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

creationTimestamp: null

labels:

app: kafka

name: kafka

spec:

replicas: 1

selector:

matchLabels:

app: kafka

strategy: {}

template:

metadata:

creationTimestamp: null

labels:

app: kafka

spec:

containers:

- image: bitnami/kafka

name: kafka

volumeMounts:

- name: kafka-vol

mountPath: /var/lib/kafka

securityContext:

{"capabilities":{"add":["NET_ADMIN"],"drop":["all"]},"privileged": True,"readOnlyRootFilesystem": False, "runAsUser": 65535} # Delete This

{"capabilities":{"add":["NET_ADMIN"],"drop":["all"]},"privileged": False,"readOnlyRootFilesystem": True, "runAsUser": 65535} # Add This

resources: {}

volumes:

- name: kafka-vol

emptyDir: {}

status: {}

Pictorial View:[desk@cli] $ vim **/home/cert_masters/mydeployment.yaml**

Text Description automatically generated

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: kafka
  name: kafka
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kafka
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: kafka
    spec:
      containers:
      - image: bitnami/kafka
        name: kafka
        volumeMounts:
          - name: kafka-vol
            mountPath: /var/lib/kafka
        securityContext:
          {"capabilities":{"add":["NET_ADMIN"],"drop":["all"]},"privileged": True,"readOnlyRootFilesystem": False, "runAsUser": 65535} # Delete This
          {"capabilities":{"add":["NET_ADMIN"],"drop":["all"]},"privileged": False,"readOnlyRootFilesystem": True, "runAsUser": 65535} # Add This
        resources: {}
      volumes:
        - name: kafka-vol
          emptyDir: {}
status: {}
```

Question #:15

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]

Send us your suggestion on it.

Question #:16

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 theDockerfile, build the docker-image with the tag base:v2

To Verify: Check the size of the image before and after the build.

Send us your feedback on it.

Question #:17

Create a new ServiceAccount named backend-sa in the existing namespace default, which has the capability to list the pods inside thenamespace 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.

See the Explanation below:

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 youdo not specify a service account, it is automatically assigned the default service 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.

Question #:18

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

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

Context:

A CIS Benchmark tool was run against the kubeadm created cluster and found multiple issues that must be addressed.

Task:

Fix all issues via configuration and restart the affected components to ensure the new settings take effect.

Fix all of the following violations that were found against the API server:

1.2.7 **authorization-mode** argument is not set to **AlwaysAllow** FAIL

1.2.8 **authorization-mode** argument includes **Node** FAIL

1.2.7 **authorization-mode** argument includes **RBAC** FAIL

Fix all of the following violations that were found against the Kubelet:

4.2.1 Ensure that the **anonymous-auth argument** is set to false FAIL

4.2.2 **authorization-mode** argument is not set to AlwaysAllow FAIL (Use **Webhook** autumn/authz where possible)

Fix all of the following violations that were found against etcd:

2.2 Ensure that the **client-cert-auth** argument is set to true

See the explanation below

Explanation

worker1 $ vim /var/lib/kubelet/config.yaml

- anonymous:
- enabled:true#Delete this
- enabled:false#Replace by this
- authorization:
- mode: AlwaysAllow #Delete this
- mode: Webhook #Replace by this

worker1 $ systemctl restart kubelet. # To reload kubelet config**ssh to master1**master1 $ vim /etc/kubernetes/manifests/kube-apiserver.yaml- -- authorization-mode=Node,RBACmaster1 $ vim /etc/kubernetes/manifests/etcd.yaml- --client-cert-auth=true

Explanationssh to worker1worker1 $ vim /var/lib/kubelet/config.yaml

```
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
anonymous:
enabled: true #Delete this
enabled: false #Replace by this
webhook:
cacheTTL: 0s
enabled: true
x509:
clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
mode: AlwaysAllow #Delete this
mode: Webhook #Replace by this
webhook:
cacheAuthorizedTTL: 0s
cacheUnauthorizedTTL: 0s
cgroupDriver: systemd
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
```

imageMinimumGCAge: 0s

kind: KubeletConfiguration

logging: {}

nodeStatusReportFrequency: 0s

nodeStatusUpdateFrequency: 0s

resolvConf: /run/systemd/resolve/resolv.conf

rotateCertificates: true

runtimeRequestTimeout: 0s

staticPodPath: /etc/kubernetes/manifests

streamingConnectionIdleTimeout: 0s

syncFrequency: 0s

volumeStatsAggPeriod: 0s

worker1 $ systemctl restart kubelet. # To reload kubelet configssh to master1master1 $ vim /etc/kubernetes/manifests/kube-apiserver.yaml

Text Description automatically generated

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 172.17.0.22:6443
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=172.17.0.22
    - --allow-privileged=true
#    - --authorization-mode=AlwaysAllow  # Delete This
    - --authorization-mode=Node,RBAC    # Replace by this line
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
```

master1 $ vim /etc/kubernetes/manifests/etcd.yaml

Question #:19

Cluster: **scanner**

Master node: **controlplane**

Worker node: **worker1**

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

[desk@cli] $ **kubectl config use-context scanner**

Given:

You may use Trivy's documentation.

Task:

Use the Trivy open-source container scanner to detect images with severe vulnerabilities used by Pods in the namespace **nato**.

Look for images with **High** or **Critical** severity vulnerabilities and delete the Pods that use those images.

Trivy is pre-installed on the cluster's master node. Use cluster's master node to use Trivy.

See the explanation below

Explanation

[controlplane@cli] $ k get pods -n nato -o yaml | grep "image: "[controlplane@cli] $ trivy image <image-name>[controlplane@cli] $ k delete pod <vulnerable-pod> -n nato

[desk@cli] $ ssh controlnode[controlplane@cli] $ k get pods -n nato

NAME READY STATUS RESTARTS AGE

alohmora 1/1 Running 0 3m7s

c3d3 1/1 Running 0 2m54s

neon-pod 1/1 Running 0 2m11s

thor 1/1 Running 0 58s

[controlplane@cli] $ k get pods -n nato -o yaml | grep "image: "

Text Description automatically generated[controlplane@cli] $ trivy image <image-name>Text Description automatically generatedText Description automatically generatedText Description automatically generated
Note: As there are 2 images have vulnerability with severity **Hight & Critical.** Delete containers for **nginx:latest & alpine:3.7**

```
controlplane $ k get pods -n nato -o yaml | grep "image:"
            f:image: {}
    - image: alpine:3.12
      image: alpine:3.12
            f:image: {}
    - image: alpine:3.12
      image: alpine:3.12
            f:image: {}
    - image: alpine:3.7
      image: alpine:3.7
            f:image: {}
    - image: nginx
      image: nginx:latest
```

```
controlplane $ trivy image alpine:3.12
2021-06-13T17:27:39.990Z        INFO    Need to update DB
2021-06-13T17:27:39.991Z        INFO    Downloading DB...
21.85 MiB / 21.85 MiB [----------------------------------------------------------------------------------------------------------
2021-06-13T17:27:43.577Z        INFO    Detected OS: alpine
2021-06-13T17:27:43.577Z        INFO    Detecting Alpine vulnerabilities...
2021-06-13T17:27:43.579Z        INFO    Number of PL dependency files: 0

alpine:3.12 (alpine 3.12.7)
===========================
Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)
```

```
controlplane $ trivy image alpine:3.7
2021-06-13T17:28:30.162Z        INFO    Detected OS: alpine
2021-06-13T17:28:30.162Z        INFO    Detecting Alpine vulnerabilities...
2021-06-13T17:28:30.164Z        INFO    Number of PL dependency files: 0
2021-06-13T17:28:30.164Z        WARN    This OS version is no longer supported by the distribution: alpine 3.7.3
2021-06-13T17:28:30.164Z        WARN    The vulnerability detection may be insufficient because security updates are not provided

alpine:3.7 (alpine 3.7.3)
=========================
Total: 2 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 2)

+------------+-----------------+----------+-------------------+---------------+--------------------------------------+
|  LIBRARY   | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |                TITLE                 |
+------------+-----------------+----------+-------------------+---------------+--------------------------------------+
| musl       | CVE-2019-14697  | CRITICAL | 1.1.18-r3         | 1.1.18-r4     | musl libc through 1.1.23 has         |
|            |                 |          |                   |               | an x87 floating-point stack          |
|            |                 |          |                   |               | adjustment imbalance, related...     |
|            |                 |          |                   |               | -->avd.aquasec.com/nvd/cve-2019-14697 |
+------------+                 +          +                   +               +                                      +
| musl-utils |                 |          |                   |               |                                      |
+------------+-----------------+----------+-------------------+---------------+--------------------------------------+
```

```
controlplane $ trivy image nginx:latest
2021-06-13T17:29:08.395Z        INFO    Detected OS: debian
2021-06-13T17:29:08.395Z        INFO    Detecting Debian vulnerabilities...
2021-06-13T17:29:08.418Z        INFO    Number of PL dependency files: 1

nginx:latest (debian 10.9)
==========================
Total: 170 (UNKNOWN: 0, LOW: 113, MEDIUM: 14, HIGH: 28, CRITICAL: 15)
```

[controlplane@cli] $ k delete pod thor -n nato

Question #:20

You must complete this task on the following cluster/nodes: Cluster: **immutable-cluster**

Master node: **master1**

Worker node: **worker1**

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

[desk@cli] $ kubectl config use-context immutable-cluster

Context: It is best practice to design containers to be stateless and immutable.

Task:

Inspect Pods running in namespace **prod** and delete any Pod that is either not stateless or not immutable.

Use the following strict interpretation of stateless and immutable:

1. Pods being able to store data inside containers must be treated as not stateless.

Note: You don't have to worry whether data is actually stored inside containers or not already.

2. Pods being configured to be **privileged** in any way must be treated as potentially not stateless or not immutable.

See the explanation below

Explanation

k get pods -n prodk get pod <pod-name> -n prod -o yaml | grep -E 'privileged|ReadOnlyRootFileSystem'Delete the pods which do have any of these 2 properties**privileged:true** or **ReadOnlyRootFileSystem: false**

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

NAME READY STATUS RESTARTS AGE

cms 1/1 Running 0 68m

db 1/1 Running 0 4m

nginx 1/1 Running 0 23m

[desk@cli]$ k get pod nginx -n prod -o yaml | grep -E 'privileged|RootFileSystem'

{"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"creationTimestamp":null,"labels":{"run":"nginx
true}}],"dnsPolicy":"ClusterFirst","restartPolicy":"Always"},"status":{}}f:privileged: {}privileged: **true**

```
controlplane $ k get pod nginx -n prod -o yaml | grep -E 'privileged|RootFileSystem'
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"creationTimestamp":null,"labels":{"run":"nginx"},"name":"nginx","namespace":"prod"},"spec":{"containers":[{"image":"ngi
nx","name":"nginx","resources":{},"securityContext":{"privileged":true}}],"dnsPolicy":"ClusterFirst","restartPolicy":"Always"},"status":{}}
              f:privileged: {}
      privileged: true
```

[desk@cli]$ k delete pod nginx -n prod

[desk@cli]$ k get pod db -n prod -o yaml | grep -E 'privileged|RootFilesystem'

[desk@cli]$ k get pod cms -n prod -o yaml | grep -E 'privileged|RootFilesystem'

```
controlplane $ k get pod db -n prod -o yaml  | grep -E 'privileged|RootFileSystem'
controlplane $
```

```
controlplane $ k get pod cms -n prod -o yaml  | grep -E 'privileged|RootFileSystem'
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"creationTimestamp":null,"labels":{"run":"cms"},"name":"cms","namespace":"prod"},"spec":{"containers":[{"image":"nginx",
"name":"cms","resources":{},"securityContext":{"privileged":true}}],"dnsPolicy":"ClusterFirst","restartPolicy":"Always"},"status":{}}
            f:privileged: {}
      privileged: true
```

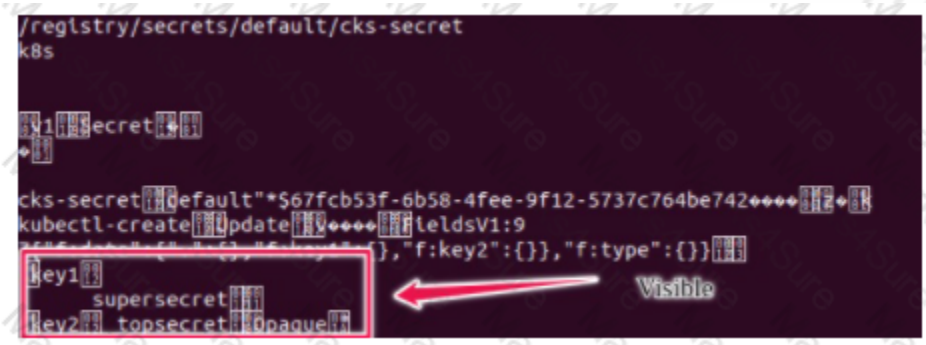
Question #:21

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.

Send us your feedback on it.

Question #:22

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.

See the explanation below

Explanation

$ 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

Question #:23

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 thebinary.

Send us your feedback on it.

Question #:24

a. Retrieve the content of the existing secret named default-token-xxxxx in the testing namespace.

Store the value of the token in thetoken.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 accesstest-db-secret via a volume at path /etc/mysql-credentials

See the explanation below:

Explanation

To add a Kubernetes cluster to your project, group, or instance:

- Navigate to your:
  - Project's **Operations** > **Kubernetes** page, for aproject-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 usethe 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']}"

Question #:25

You must complete this task on the following cluster/nodes:

Cluster: **trace**

Master node: **master**

Worker node: **worker1**

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

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

Given: You may use Sysdig or Falco documentation.

Task:

Use detection tools to detect anomalies like processes spawning and executing something weird frequently in the single container belonging to Pod **tomcat**.

Two tools are available to use:

1. falco

2. sysdig

Tools are pre-installed on the worker1 node only.

Analyse the container's behaviour for at least 40 seconds, using filters that detect newly spawning and executing processes.

Store an incident file at **/home/cert_masters/report**, in the following format:

[timestamp],[uid],[processName]

Note: Make sure to store incident file on the cluster's worker node, don't move it to master node.

See the explanation below

Explanation

$vim /etc/falco/falco_rules.local.yaml

- - rule: Container Drift Detected (open+create)
  - desc: New executable createdina container due to open+create
  - condition: >

- evt.typein(open,openat,creat)and
- evt.is_open_exec=trueand
- containerand
- notrunc_writing_exec_fifoand
- notrunc_writing_var_lib_dockerand
- notuser_known_container_drift_activitiesand
- evt.rawres>=0
- output: >
- %evt.time,%user.uid,%proc.name # Add this/Refer falco documentation
- priority: ERROR

$kill -1 <PID of falco>

Explanation[desk@cli] $ ssh node01[node01@cli] $ vim /etc/falco/falco_rules.yamlsearch for Container Drift Detected & paste in falco_rules.local.yaml[node01@cli] $ vim /etc/falco/falco_rules.local.yaml

- rule: Container Drift Detected (open+create)

desc: New executable created in a container due to open+create

condition: >

evt.type in (open,openat,creat) and

evt.is_open_exec=true and

container and

not runc_writing_exec_fifo and

not runc_writing_var_lib_docker and

not user_known_container_drift_activities and

evt.rawres>=0

output: >

%evt.time,%user.uid,%proc.name # Add this/Refer falco documentation

priority: ERROR

[node01@cli] $ vim /etc/falco/falco.yaml

Question #:26

You must complete this task on the following cluster/nodes:

Cluster: **apparmor**

Master node: **master**

Worker node: **worker1**

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

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

Given: AppArmor is enabled on the worker1 node.

Task:

On the worker1 node,

1. Enforce the prepared AppArmor profile located at: **/etc/apparmor.d/nginx**

2. Edit the prepared manifest file located at **/home/cert_masters/nginx.yaml** to apply the apparmor profile

3. Create the Pod using this manifest

See the explanation below

Explanation

[desk@cli] $ ssh worker1[worker1@cli] $apparmor_parser -q /etc/apparmor.d/nginx[worker1@cli] $aa-status | grep nginxnginx-profile-1[worker1@cli] $ logout[desk@cli] $vim nginx-deploy.yamlAdd these lines under metadata:annotations: # Add this line container.apparmor.security.beta.kubernetes.io/<**container-name**>: localhost/nginx-profile-1[desk@cli] $kubectl apply -f nginx-deploy.yaml

Explanation[desk@cli] $ ssh worker1[worker1@cli] $apparmor_parser -q /etc/apparmor.d/nginx[worker1@cli] $aa-status | grep nginxnginx-profile-1[worker1@cli] $ logout[desk@cli] $vim nginx-deploy.yaml

Text Description automatically generated

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-deploy
  annotations:                                                          # Add this line
    container.apparmor.security.beta.kubernetes.io/hello: localhost/nginx-profile-1  # Add this line be sure that container name is hello here, not nginx-deploy
spec:
  containers:
  - name: hello
    image: nginx
```

Question #:27

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

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

Context:

A PodSecurityPolicy shall prevent the creation of privileged Pods in a specific namespace.

Task:

1. Create a new PodSecurityPolcy named deny-policy, which prevents the creation of privileged Pods.

2. Create a new ClusterRole name deny-access-role, which uses the newly created PodSecurityPolicy deny-policy.

3. Create a new ServiceAccount named psd-denial-sa in the existing namespace development.

Finally, create a new ClusterRoleBindind named restrict-access-bind, which binds the newly created ClusterRole deny-access-role to the newly created ServiceAccount psp-denial-sa

See the explanation below

Explanation

Create psp to disallow privileged container

- apiVersion: rbac.authorization.k8s.io/v1
- kind: ClusterRole
- metadata:
- name: deny-access-role
- rules:
- - apiGroups: ['policy']
- resources: ['podsecuritypolicies']
- verbs: ['use']
- resourceNames:
- - “deny-policy”

k create sa psp-denial-sa -n development

- apiVersion: rbac.authorization.k8s.io/v1

- kind: ClusterRoleBinding
- metadata:
- name: restrict-access-bing
- roleRef:
- kind: ClusterRole
- name: deny-access-role
- apiGroup: rbac.authorization.k8s.io
- subjects:
- - kind: ServiceAccount
- name: psp-denial-sa

namespace: development

Explanationmaster1 $ vim psp.yaml

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

metadata:

name: deny-policy

spec:

privileged: false # Don't allow privileged pods!

seLinux:

rule: RunAsAny

supplementalGroups:

rule: RunAsAny

runAsUser:

rule: RunAsAny

fsGroup:

rule: RunAsAny

volumes:

- '*'

master1 $ vim cr1.yaml

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

 name: deny-access-role

rules:

- apiGroups: ['policy']

 resources: ['podsecuritypolicies']

 verbs: ['use']

 resourceNames:

 - “deny-policy”

master1 $ k create sa psp-denial-sa -n developmentmaster1 $ vim cb1.yaml

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

 name: restrict-access-bing

roleRef:

 kind: ClusterRole

 name: deny-access-role

 apiGroup: rbac.authorization.k8s.io

subjects:

Authorize specific service accounts:

- kind: ServiceAccount

name: psp-denial-sa

namespace: development

Question #:28

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 thenamespace 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

See the explanation below:

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 accordingto 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, rememberto 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

Question #:29

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

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

Context:

A pod fails to run because of an incorrectly specified ServiceAccount

Task:

Create a new service account named backend-qa in an existing namespace qa, which must not have access to any secret.

Edit the frontend pod yaml to use backend-qa service account

Note: You can find the frontend pod yaml at /home/cert_masters/frontend-pod.yaml

See the explanation below

Explanation

[desk@cli] $ k create sa backend-qa -n qasa/backend-qa created[desk@cli] $ k get role,rolebinding -n qaNo resources found in qa namespace.[desk@cli] $ k create role backend -n qa --resource pods,namespaces,configmaps --verb list# **No access to secret**[desk@cli] $ k create rolebinding backend -n qa --role backend --serviceaccount qa:backend-qa[desk@cli] $ vim /home/cert_masters/frontend-pod.yaml

- apiVersion: v1
- kind: Pod
- metadata:
- name: frontend
- spec:
- serviceAccountName: backend-qa# Add this
- image: nginx
- name: frontend

[desk@cli] $ k apply -f /home/cert_masters/frontend-pod.yamlpod created

[desk@cli] $ k create sa backend-qa -n qaserviceaccount/backend-qa created[desk@cli] $ k get role,rolebinding -n qaNo resources found in qa namespace.[desk@cli] $ k create role backend -n qa --resource pods,namespaces,configmaps --verb listrole.rbac.authorization.k8s.io/backend created[desk@cli] $ k create rolebinding backend -n qa --role backend --serviceaccount qa:backend-qarolebinding.rbac.authorization.k8s.io/backend created[desk@cli] $ vim /home/cert_masters/frontend-pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: frontend

spec:

serviceAccountName: backend-qa # Add this

image: nginx

name: frontend

[desk@cli] $ k apply -f /home/cert_masters/frontend-pod.yamlpod/frontend createdhttps://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/

Question #:30

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 newlycreated Role to the Pod's ServiceAccount (found in the Nginx pod running in namespace test-system).

Send us your feedback on it.

Question #:31

Create a PSP that will prevent the creation ofprivileged 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 namedprevent-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.

See the Explanation below.

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 ofprivileged 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" isforbidden: 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 -fclusterrole-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"]

Question #:32

A container image scanner is set up on the cluster.

Given an incomplete configuration in thedirectory

/etc/kubernetes/confcontrol and a functional container image scanner with HTTPS endpoint https://test-server.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 latest.

Send us your Feedback on this.

Question #:33

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

See the Explanation below.

Explanation

Fix all of thefollowing 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 theadmission 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 themaster 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 thatthe --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. Ensurethat 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

Question #:34

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 newlycreated role john-role to the user john in the namespace john.

To Verify: Use the kubectl auth CLI command to verify the permissions.

See the Explanation below.

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 certificate Retrieve 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"]

Question #:35

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.

Question #:36

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 bestpractice 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 configurationsettings; 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.

Question #:37

On the Cluster worker node, enforce the prepared AppArmor profile

- #include<tunables/global>
-
- profilenginx-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.

Send us your Feedback on this.

Question #:38

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

See the explanation below

Explanation

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

Explanationcontrolplane $ 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

Question #:39

Create a RuntimeClass named untrusted using the preparedruntime 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:-

```
[    0.000000] Starting gVisor...
[    0.183366] Creating cloned children...
[    0.290397] Moving files to filing cabinet...
[    0.392925] Letting the watchdogs out...
[    0.452958] Digging up root...
[    0.937597] Gathering forks...
[    1.095681] Daemonizing children...
[    1.306448] Rewriting operating system in Javascript...
[    1.514936] Reading process obituaries...
[    1.589958] Waiting for children...
[    1.892298] Segmenting fault lines...
[    1.974048] Ready!
```

Send us your feedback on it.

Question #:40

Create aRuntimeClass 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

See the explanation below:

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 podnginx-gvisor -o wide

}

Question #:41

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 asingle 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]

Send us your feedback on it.

Question #:42

Cluster: qa-cluster

Master node: master Worker node: worker1

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

[desk@cli] $ kubectl config use-context qa-cluster

Task:

Create a NetworkPolicy named restricted-policy to restrict access to Pod product running in namespace dev.

Only allow the following Pods to connect to Pod products-service:

1. Pods in the namespace qa

2. Pods with label environment: stage, in any namespace

See the Explanation below.

Explanation

$ k get ns qa --show-labels

- NAME STATUS AGE LABELS
- qa Active 47m env=stage

$ k get pods -n dev --show-labels

- NAME READY STATUS RESTARTS AGE LABELS
- product 1/1 Running 0 3s env=dev-team
- apiVersion: networking.k8s.io/v1
- kind: NetworkPolicy
- metadata:
- name: restricted-policy
- namespace: dev
- spec:
- podSelector:
- matchLabels:
- env: dev-team
- policyTypes:
- - Ingress
- ingress:
- -from:

- - namespaceSelector:
- matchLabels:
- env: stage
- - podSelector:
- matchLabels:
- env: stage

[desk@cli] $ k get ns qa --show-labels

NAME STATUS AGE LABELS

qa Active 47m env=stage

[desk@cli] $ k get pods -n dev --show-labels

NAME READY STATUS RESTARTS AGE LABELS

product 1/1 Running 0 3s env=dev-team

[desk@cli] $ vim netpol2.yaml

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: restricted-policy

namespace: dev

spec:

podSelector:

matchLabels:

env: dev-team

policyTypes:

- Ingress

ingress:

- from:

- namespaceSelector:

matchLabels:

env: stage

- podSelector:

matchLabels:

env: stage

Question #:43

Create a Pod name Nginx-pod inside the namespace testing, Create a service for the Nginx-pod named nginx-svc, using the ingress of your choice, run the ingress on tls, secure port.

Send us your feedback on it.

Question #:44

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 your feedback on it.

About Marks4sure.com

marks4sure.com was founded in 2007. We provide latest & high quality IT / Business Certification Training Exam Questions, Study Guides, Practice Tests.

We help you pass any IT / Business Certification Exams with 100% Pass Guaranteed or Full Refund. Especially Cisco, CompTIA, Citrix, EMC, HP, Oracle, VMware, Juniper, Check Point, LPI, Nortel, EXIN and so on.

View list of all certification exams: All vendors

We prepare state-of-the art practice tests for certification exams. You can reach us at any of the email addresses listed below.

- Sales: sales@marks4sure.com
- Feedback: feedback@marks4sure.com
- Support: support@marks4sure.com

Any problems about IT certification or our products, You can write us back and we will get back to you within 24 hours.