CKS certification - CKS 2021 latest true - Exercise questions 03

tags: K8S certification CKS 2021 latest true question kubernetes Pod Security SecurityPolicy audit-policy

CKS certification - CKS 2021 latest true - Exercise questions 03

1 Mirror Scan ImagePolicyWebhook
- Topic Overview
- Analyze

2 Sysdig & FALOC detection POD
- Topic Overview
- Analyze

3 ClusterRole
- Topic Overview
- Analyze

4 AppArmor
- Topic Overview
- Analyze

5 PodSecurityPolicy
- Topic Overview
- Analyze

6 Network Policy NetworkPolicy

Topic Overview

Analyze

7 DockerFile test

Topic Overview

Analyze

8 POD security

Topic Overview

Analyze

9 Create serviceAccount

Topic Overview

Analyze

10 trivy detection mirror safety

Topic Overview

Analyze

11 Create a secret

Topic Overview

Analyze

12 kube-bench

Topic Overview

Analyze

13 gVisor

Topic Overview

Analyze

14 audit

Topic Overview

Analyze

15 default network strategy

Topic Overview

Analyze

16 Modify the API Server parameter

TOP

Topic Overview

Analyze

illustrate

At the beginning of June, the author isCKS certification - CKS 2021 latest Zhenti - Exercise question 01 with CKS certification - CKS 2021 latest Zhenti - Exercise Question 02 Two sets of CKS true questions were shared in the middle, because the time was more rudimentary. Recently, the enthusiastic group feedback has given the heart of the CKS test, sharing a truth of the subject, and the author put it properly, and put it here for everyone to learn!

The purpose of this article:
One is to provide you with the common test of CKS to facilitate everyone to study;
Second, give you a place to share and communicate. You are welcome to retention in the comment area;

The author has already registered CKS in mid-November. If you finish it, you will finish the latest true analysis, welcome to prepare for CKA, CKS's attention.K8S certification with K8S & Docker (Reproduces the Punroups in the Bowen contains CKA and CKS review points, as well as the experiments made by the author, the main difference is that there is no form of "series of blog posts).

1 Mirror Scan ImagePolicyWebhook

Topic Overview

```
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 scall scan for and reject the use of vulnerable images.
task
You have to complete the entire task on the cluster master node,where all services
Glven an incomplete configuration in directory /etc/kubernetes/aa and a functional

1.enable the necessary plugins to create an image policy
2.validate the control configuration and chage it to an implicit deny
3.Edit the configuration to point the provied HTTPS endpoint correctiy

Finally,test if the configurateion is working by trying to deploy the valnerable re
```

TOP

Analyze

Need from the console SSH to the Master node, edit /etc/kubernetes/manifest/kube-apiServer.yaml Quote ImagePolicyWebhook from the file:

```
- --enable-admission-plugins=NodeRestriction,ImagePolicyWebhook
- --admission-control-config-file=/etc/kubernetes/aa/admission_configuration.json
```

Configure HostPath:

```
volumes:
- hostPath:
    path: /etc/kubernetes/aa/
  name: xxx
```

Configure VolumeMounts:

```
volumeMounts:
- mountPath: /etc/kubernetes/aa/
  name: xxx
  # There will be a readonly: true when the exam is: True, delete this line
```

Edit Admission_Configuration.json (the title will be given), modify the defaultAllow for false:

```
{
  "imagePolicy": {
     "kubeConfigFile": "/etc/kubernetes/aa/kubeconfig.yaml",
     "allowTTL": 50,
     "denyTTL": 50,
     "retryBackoff": 500,
     "defaultAllow": false  # Change to False
  }
}
```

Edit /etc/kubernetes/aa/kubeconfig.yaml, add a WebHook Server address:

```
apiVersion: v1
kind: Config
clusters:
- cluster:
    certificate-authority: /etc/kubernetes/aa/webhook.pem
```

TOP

```
    server: http://192.168.26.60:1323/image_policy  # Add WebHook Server Address
  name: bouncer_webhook
contexts:
- context:
    cluster: bouncer_webhook
    user: api-server
  name: bouncer_validator
current-context: bouncer_validator
preferences: {}
users:
- name: api-server
  user:
    client-certificate: /etc/kubernetes/aa/apiserver-client.pem
    client-key:  /etc/kubernetes/aa/apiserver-clientkey.pem
```

Restart kubelet:

```
systemctl restart kubelet
kubectl apply -f /cks/1/web1.yaml
```

Verify that the image of Latest is not allowed:

```
root@vms61:/etc/kubernetes/aa# kubectl run pod1 --image=nginx
Error from server (Forbidden): pods "pod1" is forbidden: image policy webhook backe
```

Reference documentation:
https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers

2 Sysdig & FALOC detection POD

Topic Overview

you may user you brower to open one additonal tab to access sysdig documentation ro Falco documentaion
Task:
user runtime detection tools to detect anomalous processes spawning and executing frequently in the sigle container belorging to Pod redis.
Tow tools are avaliable to use:
sysdig or falico

the tools are pre-installed on the cluster worker node only;the are not avaliable on the base system or the master node.
using the tool of you choice(including any non pre-install tool) analyse the container behaviour for at lest 30 seconds, using filers that detect newly spawing and executing processes.
store an incident file at /opt/2/report,containing the detected incidents one per line in the follwing format:
[timestamp],[uid],[processName]

Analyze

From the console SSH to the Worker node, first find the container ID of the container:

```
root@vms62:~# docker ps | grep redis
5ae46a497d05   dc4395f73f8d                                          "docker-entrypoi
6b715c0fea71   registry.aliyuncs.com/google_containers/pause:3.2   "/pause"
```

Scan container 30s via SYSDIG and output to the specified file:

```
# sysdig -l View Help
sysdig -M 30  -p "*%evt.time,%user.uid,%proc.name" container.id=5ae46a497d05 > /opt
```

Reference documentation:
docs.sysdig.com
blog/2015/11/monitoring-kubernetes-with-sysdig/

3 ClusterRole

Topic Overview

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 monitoring. Edit the existing Role bound to the Pod’s serviceAccount sa-dev-1 to only allow performing list operations, only on resources of type Endpoints.
create a new Role named role-2 in the namespace monitoring, which only allows performing update

operations, only on resources of type persistentvolumeclaims.
create a new RoleBinding named role-2-binding binding the newly created Role to the Pod's serviceAccount.
Don't delete the existing RoleBinding.

Analyze

Modify the role permissions of SA-DEV-1, only allowing the endpoints to do LIST operations.
View RoleBindings Sa-dev-1 Role to Role-1

```
root@vms60:/cks/9# kubectl get rolebindings -n monitoring
NAME       ROLE          AGE
sa-dev-1   Role/role-1   7d16h
```

Edit Role-1 Permissions: `kubectl edit role role-1 -n monitoring`

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  creationTimestamp: "2021-01-22T16:48:36Z"
  name: role-1
  namespace: monitoring
  resourceVersion: "9528"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/monitoring/roles/role-1
  uid: 0dd5f94d-c27d-4052-a036-12c6c1006858
rules:
- apiGroups:
  - ""
  resources:
  - endpoints  # Only Allow Endpoints Resource List
  verbs:
  - list
```

Create Role named Role-2 and bind SA-DEV-1 through rolebinding, only allowing the Update operation for PersistentVoluMeclaims.

```
kubectl create role role-2 --resource=persistentvolumeclaims --verb=update -n monit
kubectl create rolebinding role-2-binding --role=role-2 --serviceaccount=monitoring
```

TOP

Reference documentation:
docs/reference/access-authn-authz/rbac/

4 AppArmor

Topic Overview

Context
AppArmor is enabled on the cluster worker node. An AppArmor profile is prepared, but not enforced yet.
You may use your browser to open one additional tab to access theAppArmor documentation.
Task
On the cluster worker node, enforce the prepared AppArmor profile located at /etc/apparmor.d/nginx_apparmor . Edit the prepared manifest file located at /cks/4/pod1.yaml to apply the AppArmor profile.
Finally, apply the manifest file and create the pod specified in it

Analyze

Need from the console SSH to the Worker node.
Execute the AppArmor Policy Module:

```
#No n't GREP to the instructions
apparmor_status | grep nginx-profile-3
#
apparmor_parser -q nginx_apparmor
root@vms62:/etc/apparmor.d# apparmor_status | grep nginx
   nginx-profile-3
```

Create a POD to add annotations:

```
apiVersion: v1
kind: Pod
metadata:
  name: podx
  # Add ANNOTATIONS, PODX name, and the name of the Container, nginx-profile-3 is t
  annotations:
     container.apparmor.security.beta.kubernetes.io/podx: localhost/nginx-profile-3
spec:
  containers:
  - image: nginx:1.9
```

```
    imagePullPolicy: IfNotPresent
    name: podx
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}

 Final execution:
kubectl apply -f /cks/4/pod1.yaml
```

Reference documentation:
docs/tutorials/clusters/apparmor/

5 PodSecurityPolicy

Topic Overview

context
A PodsecurityPolicy shall prevent the creati on of privileged Pods in a specific namespace.
Task
Create a new PodSecurityPolicy named prevent-psp-policy , which prevents the creation of privileged Pods.
Create a new ClusterRole named restrict-access-role , which uses the newly created PodSecurityPolicy prevent psp-policy .
Create a new serviceAccount named psp-denial-sa in the existing namespace development .
Finally, create a new clusterRoleBinding named dany-access-bind , which binds the newlycreated ClusterRole restrict-access-role to the newly created serviceAccount

Analyze

1. Create a PodSecurityPolicy called Prevent-PSP-Policy to block the creation of Privileged Pod
2. Creating a CLUSTERROLE named Restrict-Access-Role allows newly created PRSecurityPolicy.
3. Create a serviceAccount called PSP-Denial-Sa in the development namespace. Create ClusterRoleBinding named Dany-Access-Bind, binding serviceAccount and ClusterRole that just created.

You need to enable PodSecurityPolicy from the console ssh to the master node to ensure that PodSecurityPolicy is enabled /etc/kubiserver.yaml. (Enabled in the exam)

```
- --enable-admission-plugins=NodeRestriction,PodSecurityPolicy
```

Create a PODSecurityPolicy called Prevent-PSP-Policy to block the creation of Privileged Pod:

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: prevent-psp-policy
spec:
  privileged: false  #false indicates that the POD is prohibited from creating priv
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  volumes:
  - '*'
```

Create ServiceAccount and Cluserrole and bind to ClusterRoleBing:

```
kubectl create clusterrole restrict-access-role --verb=use --resource=psp --resourc
kubectl create sa psp-denial-sa -n development
kubectl create clusterrolebinding dany-access-bind --clusterrole=restrict-access-rc
```

Reference documentation:
docs/concepts/policy/pod-security-policy/

6 Network Policy NetworkPolicy

Topic Overview

create a NetworkPolicy named pod-access torestrict access to Pod products-service running in namespace development .

TOP

only allow the following Pods to connect to Pod products-service :
Pods in the namespace testing
Pods with label environment: staging , in any namespace
Make sure to apply the NetworkPolicy. You can find a skelet on manifest file at /cks/6/p1.yaml

Analyze

In the development namespace, create a NetworkPolicy called POD-Access's POD called Products-Service, only allows the namespace to be TEST's POD or an Environment: Staging tag POD in any namespace. Access.

If POD or Namespace has no label, you can play tags:

```
kubectl label ns testing name=testing
kubectl label pod products-service environment=staging
```

Create NetworkPolicy:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: pod-access
  namespace: development
spec:
  podSelector:
    matchLabels:
       environment: staging
  policyTypes:
  - Ingress
  ingress:
  - from: # Name the space has a name: Testing tag POD
    - namespaceSelector:
        matchLabels:
          name: testing
  - from:  # All namespaces have an Environment: Staging tag POD
    - namespaceSelector:
        matchLabels:
      podSelector:
        matchLabels:
           environment: staging
```

Reference documentation:
docs/concepts/services-networking/network-policies/

7 DockerFile test

Topic Overview

Task
Analyze and edit the given Dockerfile (based on the ubuntu:16.04 image) /cks/7/Dockerfile, fixing two instructions present in the file being prominent security/best-practice issues.
Analyze and edit the given manifest file /cks/7/deployment.yaml fixing two fields present in the file being prominent security/best-practice issues.

Analyze

Detect DockerFile files, there are two errors:

```
$ vim /cks/7/Dockerfile
#USER root
$ vim /cks/7/deployment.yaml
# securityContext:
#  {"Capabilities": {'add':{NET_BIND_SERVICE}, 'drop: []'}, 'privileged': TRUE}
```

```
# Two root comments
#USER root
```

Detect deployment YAML files, there are two errors:

```
Modified to: Apiversion: apps/v1
 Note:#{"Capabilities": {'add':{NET_BIND_SERVICE}, 'drop: []'}, 'privileged': TRUE}
```

Reference documentation:
docs.docker.com/develop/develop-images/dockerfile_best-practices/

8 POD security

Topic Overview

context

It is best-practice to design containers to best teless and immutable.

Task

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

use the following strict interpretation of stateless and immutable:

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

You don’t have to worry whether data is actually stored inside containers or not already. Pods being configured to be privileged in any way must be treated as potentially not stateless and not immutable.

Analyze

Get all PODs, see if there is privileged privileged or Mount Volume's POD

```
kubectl get pods NAME -n testing -o jsonpath={.spec.volumes} | jq
kubectl get pods NAME -o yaml -n testing | grep "privi.*: true"
```

Then remove privileged or Mount Volume's POD.

Reference documentation:

docs/tasks/configure-pod-container/security-context/

9 Create serviceAccount

Topic Overview

context

A Pod fails to run because of an incorrectly specified ServiceAcccount.

Task

create a new ServiceAccount named frontend-sa in the existing namespace qa ,which must not have access to any secrets.

Inspect the Pod named frontend running in the namespace qa . Edit the Pod to use the newly created serviceAccount

Analyze

Create serviceAccount frontend-sa in QA namespace, and access any seconds is not allowed. Create a POD named frontend-sa using the serviceAccount.

Create serviceAccount:

```
apiVersion: v1
kind: ServiceAccount
automountServiceAccountToken: false #
metadata:
  name: frontend-sa
  namespace: qa
```

Create a POD to use this serviceAccount

```
apiVersion: v1
kind: Pod
metadata:
  name: "frontend"
  namespace: "qa"
spec:
  serviceAccountName: "frontend-sa"
  containers:
  - image: nginx:1.9
    imagePullPolicy: IfNotPresent
    name: podx
    resources: {}
status: {}
```

Delete unused serviceAccount, the exam should have 2:

```
root@vms60:/cks/9# kubectl delete sa -n qa default
serviceaccount "default" deleted
```

Reference documentation:
docs/tasks/configure-pod-container/configure-service-account/

10 trivy detection mirror safety

Topic Overview

Task
Use the Trivy open-source container scanner to detect images with severe vulnerabilities used by Pods in the namespace yavin .
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 only; it is not available on the base system or the worker nodes. You'll have to connect to the cluster's master node to use Trivy

Analyze

Use TRIVY to scan the image of the POD in YAVIN namespace and delete the high or critical risk of POD. ** Trivy is installed on the Master node and needs to log in from the console ssh.

List the mirror of POD:

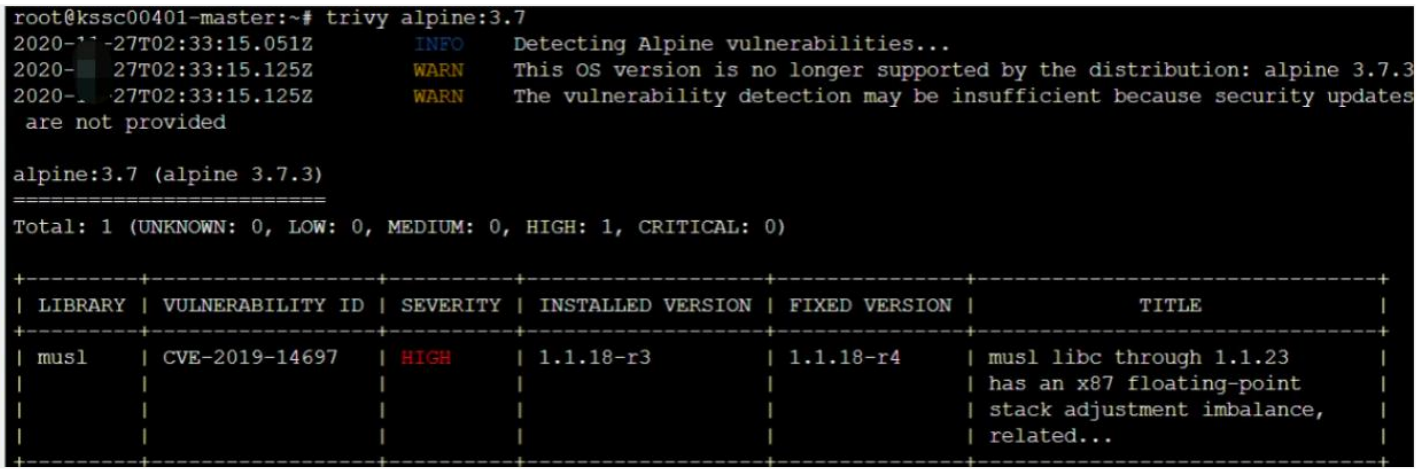
```
root@vms60:/cks/9# kubectl get pod -n yavin
NAME        READY   STATUS    RESTARTS   AGE
baby-yoda   1/1     Running   1          7d11h
r2d2        1/1     Running   2          7d11h
rex         1/1     Running   1          7d11h
yoda        1/1     Running   1          7d11h

root@vms60:/cks/9# for i in baby-yoda r2d2 rex yoda ; do
> echo $i
> kubectl get pod $i -n yavin -o yaml | grep "image: "; done
baby-yoda
            f:image: {}
    image: amazonlinux:1
    image: amazonlinux:1
r2d2
            f:image: {}
    image: amazonlinux:1
    image: amazonlinux:1
rex
            f:image: {}
    image: alpine:3.12
    image: alpine:3.12
yoda
            f:image: {}
    image: alpine:3.12
    image: alpine:3.12
```

```
trivy image --skip-update amazonlinux:1 | egrep -i "High|Critical"
```

TOP

The result of the scan is similar to the following figure, remove the POD using the image:



Reference documentation:
github.com/aquasecurity/trivy

11 Create a secret

Topic Overview

Task
Retrieve the content of the existing secret named db1-test in the istio-system namespace.
store the username field in a file named /cks/11/old-username.txt , and the password field in a file named /cks/11/old-pass.txt.
You must create both files; they don exist yet.
Do not use/modify the created files in the following steps, create new temporaryfiles if needed.
Create a new secret named test-workflow in the istio-system namespace, with the following content:

username : thanos
password : hahahaha

Finally, create a new Pod that has access to the secret test-workflow via a volume:

pod name dev-pod
namespace istio-system
container name dev-container
image nginx:1.9
volume name dev-volume
mount path /etc/test-secret

TOP

Analyze

Save DB1-Test's UserName and Password Base64 decodes to the specified file:

```
kubectl get secrets -n istio-system db1-test -o jsonpath='{.data.username}'  | base
kubectl get secrets -n istio-system db1-test -o jsonpath='{.data.password}'  | base
```

Create a Secret for Test-Workflow
username : thanos
password : hahahaha
As a key value.

```
kubectl create secret generic test-workflow --from-literal=username=thanos --from-l
```

Create a POD to use the second.

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: dev-pod
  name: dev-pod #Pod name
  namespace: istio-system #Namespaces
spec:
  volumes:
  - name: dev-volume # Create Volume
    secret:
      secretName: test-workflow
  containers:
  - image: nginx:1.9 #
    name: dev-container # Specify the container name
    resources: {}
    volumeMounts:  # Specify the mount path
    - mountPath: /etc/test-secret
      name: dev-volume
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

TOP

Reference documentation:
docs/concepts/configuration/secret/

12 kube-bench

Topic Overview

context
ACIS Benchmark tool was run against the kubeadm-created cluster and found multiple issues that must be addressed immediately.

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:

Ensure that the 1.2.7 --authorization-mode FAIL argument is not set to AlwaysAllow
Ensure that the 1.2.8 --authorization-mode FAIL argument includes Node
Ensure that the 1.2.9 --authorization-mode FAIL argument includes RBAC
Ensure that the 1.2.18 --insecure-bind-address FAIL argument is not set
Ensure that the 1.2.19 --insecure-port FAIL argument is set to 0

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

Ensure that the 4.2.1 anonymous-auth FAIL argument is set to false
Ensure that the 4.2.2 --authorization-mode FAIL argument is not set to AlwaysAllow
Use webhook authn/authz

Fix all of the following violations that were found against etcd:
Ensure that the 4.2.1 --client-cert-auth FAIL argument is set to true

Analyze

Need from the console SSH to the Master node.
1 api-server

```
#kube-bench master
1.2.7 Edit the API server pod specification file /etc/kubernetes/manifests/kube-api
on the master node and set the --authorization-mode parameter to values other than
One such example could be as below.
--authorization-mode=RBAC

```

TOP

```
1.2.8 Edit the API server pod specification file /etc/kubernetes/manifests/kube-api
on the master node and set the --authorization-mode parameter to a value that inclu
--authorization-mode=Node,RBAC

1.2.9 Edit the API server pod specification file /etc/kubernetes/manifests/kube-api
on the master node and set the --authorization-mode parameter to a value that inclu
for example:
--authorization-mode=Node,RBAC

1.2.18 Edit the API server pod specification file /etc/kubernetes/manifests/kube-ap
on the master node and remove the --insecure-bind-address parameter.

1.2.19 Edit the API server pod specification file /etc/kubernetes/manifests/kube-ap
on the master node and set the below parameter.
--insecure-port=0
```

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

```
#change into
- --authorization-mode=Node,RBAC
- --insecure-port=0
#delete
- --insecure-bind-address=0.0.0.0
```

2 kubelet

The practice environment is PASS, modified when the exam is changed /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

```
#kube-bench node
[PASS] 4.2.1 Ensure that the --anonymous-auth argument is set to false (Scored)
[PASS] 4.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllc

#Add to
Environment="KUBELET_SYSTEM_PODS_ARGS=--anonymous-auth=false"
Environment="KUBELET_SYSTEM_AUTH_ARGS=--authorization-mode=RBAC"

# #
 Addition after execstart $KUBELET_SYSTEM_PODS_ARGS $KUBELET_SYSTEM_AUTH_ARGS

#          k k

```

TOP

```
systemctl daemon-reload
systemctl restart kubelet.service
```

3 etcd

```
#kube-bench
2.2 Edit the etcd pod specification file /etc/kubernetes/manifests/etcd.yaml on the
node and set the below parameter.
--client-cert-auth="true"
```

vim /etc/kubernetes/manifests/etcd.yaml

```
#change into
- --client-cert-auth=true
```

Reference documentation:
github.com/aquasecurity/kube-bench

13 gVisor

Topic Overview

context
This cluster uses containerd as CRI runtime. Containerd default runtime handler is runc .
Containerd has been prepared to support an additional runtime handler , runsc (gVisor).

Task
Create a RuntimeClass named untrusted using the prepared runtime handler namedrunsc .
Update all Pods in the namespace client to run on gvisor, unless they are already running on anon-default runtime handler. You can find a skeleton manifest file at /cks/13/rc.yaml

Analyze

Create RuntimeClass:

```
apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:

```

```
  name: untrusted # Used to reference the name of RuntimeClass, RuntimeClass is a r
handler: runsc # The name of the corresponding CRI configuration
```

kubectl apply -f /cks/13/rc.yaml

POD references RuntimeClass, the POD is created when the test is created, modified by kubectl Edit.

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: pod
  name: nginx-gvisor
spec:
  containers:
  - image: nginx
    imagePullPolicy: IfNotPresent
    name: pod
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  runtimeClassName: untrusted
```

Reference documentation:

github.com/google/gvisor

14 audit

Topic Overview

Task

Enable audit logs in the cluster.

To do so, enable the log backend, and ensurethat:

1. logs are stored at /var/log/kubernetes/audit-logs.txt
2. log files are retained for 5 days
3. at maximum, a number of 10 auditlog files are retained

A basic policy is provided at /etc/kubernetes/logpolicy/sample-policy.yaml. it only specifies what not to log.

The base policy is located on thecluster's master node.
Edit and extend the basic policy to log:

1. namespaces changes at RequestResponse level
2. the request body of pods changes in the namespace front-apps
3. configMap and secret changes in all namespaces at the Metadata level
   Also, add a catch-all ruie to log all otherrequests at the Metadata level.
   Don't forget to apply the modifiedpolicy.

Analyze

Log in to the master node, edit the MASTER node /etc/kubistes/manifests/kube-apiServer.yaml file, add the following parameters:

```
# Define the Audit Policy YAML file location, mount HostPath
- --audit-policy-file=/etc/kubernetes/logpolicy/sample-policy.yaml
# Defines the location of the audit log, mount by HostPath
- --audit-log-path=/var/log/kubernetes/audit-logs.txt
# Define the maximum number of days of the old audit log file for 5 days
- --audit-log-maxage=5
# Define the maximum number of audit log files to be retained to 10
- --audit-log-maxbackup=10
```

Configure HostPath:

```
 volumes:
 - name: audit
    hostPath:
      path: /etc/kubernetes/logpolicy/sample-policy.yaml
      type: File
  - name: audit-log
    hostPath:
      path: /var/log/kubernetes/audit-logs.txt
      type: FileOrCreate
```

Configure VoluMouNT:

```
volumeMounts:
  - mountPath: /etc/kubernetes/logpolicy/sample-policy.yaml
```

TOP

```
    name: audit
    readOnly: true
  - mountPath: /var/log/kubernetes/audit-logs.txt
    name: audit-log
    readOnly: false
```

Configuring auditing strategy: vim /etc/kubernetes/logpolicy/sample-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:
  #the request body of pods changes in the namespace front-apps
  - level: Request
    resources:
    - group: ""
      resources: ["pods"]
    namespaces: ["front-apps"]

  # namespaces changes at RequestResponse level
  - level: RequestResponse
    resources:
    - group: ""
      resources: ["namespace"]

  # Log configmap and secret changes in all other namespaces at the Metadata level.
  - level: Metadata
    resources:
    - group: ""
      resources: ["secrets", "configmaps"]

  # A catch-all rule to log all other requests at the Metadata level.
  - level: Metadata
    omitStages:
      - "RequestReceived"
```

Restart kubelet after the configuration is completed:

```
systemctl restart kubelet
```

Reference documentation:
docs/tasks/debug-application-cluster/audit/

15 default network strategy

Topic Overview

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

Task
Create a new default-deny NetworkPolicy named denynetwork in the namespace development for all traffic of type Ingress .
The new NetworkPolicy must deny all Ingress traffic in the namespace development .
Apply the newly created default-deny NetworkPolicy to all Pods running in namespace development .
You can find a skeleton manifest file

Analyze

Create NetWorkPolicy named DenyetWork, reject all Ingress traffic in the development namespace:

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: denynetwork
  namespace: development
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```

kubectl apply -f /cks/15/p1.yaml

Reference documentation:
docs/concepts/services-networking/network-policies/

16 Modify the API Server parameter

Topic Overview

context
kubeadm was used to create the cluster used in this task.

Task
Reconfigure and restart the cluster's Kubernetes API server to ensure that only authenticated and authorized REST requests are allowed.
Make sure that the new configuration applies to any REST request, including local access.
Make sure that any configuration changes are permanent and still enforced after restarting the Kubernetes API server.

Analyze

Make sure that only certified and authorized REST requests are allowed.

Edit /etc/kubernetes/manifest/kube-apiServer.yaml, will below

```
- --authorization-mode=AlwaysAllow
- --enable-admission-plugins=AlwaysAdmit
```

change into:

```
- --authorization-mode=Node,RBAC
- --enable-admission-plugins=NodeRestriction
- --client-ca-file=/etc/kubernetes/pki/ca.crt
- --enable-bootstrap-token-auth=true
```

Reference documentation:
docs/reference/command-line-tools-reference/kube-apiserver/

illustrate

1. The current test version is 1.22 (as of 2021.11.15)
2. Reference documentation
   docs/concepts/security/

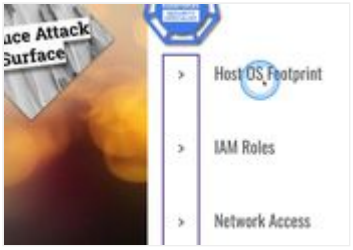
Notice!!!

- This entitled author collected by a friend who participated in the CKS test, the answer was the original reference answer; the follow-up author will organize the

latest true questions and reference answers.

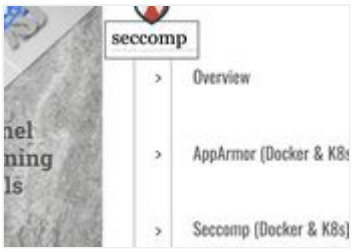
- If you have any questions about the above questions, you are welcome to discuss in the message area to facilitate a small partner study.

Intelligent Recommendation



K8S CKS 2021 [25] --- System reinforcement reduction attack surface

Article catalog 1 Introduction 2. Systemctl and Services 3. Install and investigate Services 4. Disable application on port 5. Investigate Linux Users 6. Summary 1 Introduction 2. Systemctl and Servic...



Kubernetes CKS 2021 Course【24】---System Hardening - Kernel Hardening Tools

Article catalog 1 Introduction 2. AppArmor 3. Practice - AppArmor for curl 4. Practice - AppArmor for Docker Nginx 5. Practice - AppArmor for Kubernetes Nginx 6. Seccomp 7. Practice - Seccomp for Dock...

CKS test training [1]

Article catalog 1. Test point: Use role based access Controls to minimize exposure Question 1: RBAC authorization problem Step 1: Check the current RBAC Rules Step 2: Create other RBAC Rules 2. Test p...

Kubernetes CKS 2021 Complete Course + Simulator Note [4] --- Cluster Setup - Secure Ingress

Article catalog 1 Introduction 2. Practice - create an Ingress 3. Practice - Secure an Ingress 1 Introduction 2. Practice - create an Ingress Deployment link:https://kubernetes.github.io/ingress-nginx...

Kubernetes CKS 2021 Complete Course + Simulator Note [6] --- Cluster Setup - CIS Benchmarks

Article catalog 1 Introduction 2. Practice - CIS in Action 3. Practice - kube-bench 1 Introduction 2. Practice - CIS in Action 3. Practice - kube-bench Reference link: https://github.com/aquasecurity/...

More Recommendation

Kubernetes CKS 2021 Complete Course + Simulator Note [12] --- Microservice Vulnerabilities-Manage Secrets

Extended link: Introduction Practice - Create Simple Secret Scenario Reference link:https://kubernetes.io/zh/docs/concepts/configuration/secret/#using-secrets Practice - Hack Secrets in Docker Practic...

Kubernetes CKS 2021 Course Note [13] --- Microservice Vulnerabilities - Container Runtime Sandboxes

TOP

Article catalog Introduction Practice - Container calls Linux Kernel Open Container Initiative OCI Practice - Crictl Sandbox Runtime Katacontainers Sandbox Runtime gVisor Practice - Create and use Run...

Kubernetes CKS 2021 Course【19】---Supply Chain Security - Image Vulnerability Scanning

Article catalog 1 Introduction 2. Clair and Trivy 3. Practice - Use Trivy to scan images 4. Summary 1 Introduction https://cve.mitre.org/ https://nvd.nist.gov/ 2. Clair and Trivy 3. Practice - Use Tri...

Kubernetes CKS 2021 Course【20】---Supply Chain Security - Secure Supply Chain

1 Introduction 2. Practice - Image Digest 3. Practice - Whitelist Registries with OPA 4. ImagePolicyWebhook 5. Practice - ImagePolicyWebhook...

Kubernetes CKS 2021 Course【21】---Runtime Security - Behavioral Analytics at host and container level

Article catalog 1 Introduction 2. Practice - Strace 3. Practice - Strace and /proc on ETCD 4. Practice - /proc and env variables 5. Practice - Falco and Installation 6. Practice - Use Falco to find ma...

Related Posts

- CKS certification - CKS 2021 latest Zhenti - Exercise question 02
- CKS certification - CKS 2021 latest Zhenti - Exercise question 01
- Kubernetes CKS 2021 Complete Course + Simulator Note [2] --- Network Policies
- Kubernetes CKS 2021 Course【14】---OS Level Security Domains
- Kubernetes CKS 2021 Course【15】---Microservice Vulnerabilities - mTLS
- Kubernetes CKS 2021 Course【16】---Open Policy Agent (OPA)
- Kubernetes CKS 2021 Course【17】---Supply Chain Security - Image Footprint
- Kubernetes CKS 2021 Course【18】---Supply Chain Security - Static Analysis
- Kubernetes CKS 2021 Course【23】---Runtime Security - Auditing
- Kubernetes CKS 2021 Course【22】---Runtime Security - Immutability of containers at runtime

Popular Posts

- Announcement issues after Qt development process
- CentOS6.5 Silent Mount Oracle11.2.0.4 Playing PSU Patch Steps
- Use Gaode Map API to realize historical track query
- Common operations on Python strings
- Spring boot is some of the problems encountered by the backend framework
- Video platform FPGA platform development system
- Jar package resources read file folder
- Spring system learning: day1--spring introduction
- Install pip in win10's Linux subsystem
- WeChat applet perfectly removes the scroll bar of scroll-view

Recommended Posts

- TUAK algorithm notes
- JSON object string conversion to object
- Docker entry
- Vue learning (7) v-once instruction, v-html instruction, v-pre instruction
- Vue Fetch, AXIOS, calculation properties, virtual DOM and DIFF algorithm, component development
- Regular syntax

- The reason why AOP created in SpringBoot does not take effect
- PTA 05-tree 7 path in the heap
- (02) Use of pandas library
- Design mode memo - structured

Related Tags

K8S certification

CKS certification

CKS exercise questions

CKS true question

kubernetes

docker

golang

go

kernel

nginx

TOP