背景

cks考试资格是去年活动时候跟cka一起买的 1200左右大洋吧…考了两次 ，第一次57分。我考！第二次 62分， 竟然还是没有过去…可能冥冥之中自己有所感应，今年活动的时候购买了一次机会备用的…好歹第三次算是过了86分还好…

总结一下15个题吧！

具体可参考昕光xg大佬的博客CKS认证–CKS 2021最新真题考试经验分享–练习题04。顺序记不太清了，就按照昕光xg大佬列的题目记录一下解题思路吧！墙裂推荐大佬的博客。满满的都是干货！

1. RuntimeClass gVisor

根据题目内容创建一个RuntimeClass，然后修改统一namespace下的pod

参照：官方文档https://kubernetes.io/docs/concepts/containers/runtime-class/#usage%3Cbr%3E

vim /home/cloud\_user/sandbox.yml

apiVersion: node.k8s.io/v1

kind: RuntimeClass

metadata:

name: sandbox

handler: runsc

kubectl apply -f /home/xxx/sandbox.yml

1

2

3

4

5

6

7

修改xxxx namespace下 pod的runtimeClassName: sandbox

kubectl -n xxx edit deployments.apps work01 # runtimeClassName: sandbox

kubectl -n xxx edit deployments.apps work02

kubectl -n xxx edit deployments.apps work03

1

2

3

4

基本就是这样的 修改3个deployments。增加runtimeClassName: sandbox！等待pod重建完成！

如何确定修改成功呢？

kubectl -n xxx exec xxx – dmesg

or kubectl get deployments xxx -n xxx -o yaml|grep runtime就可以吧？

2. NetworkPolicy, 限制指定pod、ns 访问指定labels的一组pod

这个题貌似一直是没有变的，网上也看了好多的解题方法 但是貌似都是有问题的

在development命名空间内，创建名为pod-access的NetworkPolicy作用于名为products-service的Pod，只允许命名空间为test的Pod或者在任何命名空间内有environment:staging标签的Pod来访问。

当然了namespace 和podname networkpolicy的名字会变

重要的是确定pod namespace的labels

kubectl get pods -n development --show-labels

kubectl get ns --show-labels

1

2

cat network-policy.yaml

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: pod-access

namespace: development

spec:

podSelector:

matchLabels:

environment: staging

policyTypes:

- Ingress

ingress:

- from: #命名空间有name: testing标签的Pod

- namespaceSelector:

matchLabels:

name: testing

- from: #所有命名空间有environment: staging标签的Pod

- namespaceSelector:

matchLabels:

podSelector:

matchLabels:

environment: staging

kubectl apply -f network-policy.yaml

官方文档参考：https://kubernetes.io/docs/concepts/services-networking/network-policies/

3. NetworkPolicy，dany all Ingress Egress

考了两次，这个题会变 但是无非是创建名为 denynetwork 的 NetworkPolicy，拒绝development命名空间内所有Ingress流量or Egress or Ingress and Egress流量

参照：https://kubernetes.io/docs/concepts/services-networking/network-policies/#default-deny-all-ingress-and-all-egress-traffic

注意审题看题目中的要求！

---

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: denynetwork

namespace： development

spec:

podSelector: {}

policyTypes:

- Ingress

---

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: denynetwork

namespace： development

spec:

podSelector: {}

policyTypes:

- Egress

---

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: denynetwork

namespace： development

spec:

podSelector: {}

policyTypes:

- Ingress

- Egress

4. trivy 检测镜像

kubectl -n development get pods

kubectl -n development get pods --output=custom-columns="NAME:.metadata.name,IMAGE:.spec.containers[\*].image"

NAME IMAGE

work1 busybox:1.33.1

work2 nginx:1.14.2

work3 amazonlinux:2

work4 amazonlinux:1

work5 centos:7

trivy image -s HIGH,CRITICAL busybox:1.33.1

trivy image -s HIGH,CRITICAL nginx:1.14.2 #HIGH and CRITICAL

trivy image -s HIGH,CRITICAL amazonlinux:2

trivy image -s HIGH,CRITICAL amazonlinux:1

trivy image -s HIGH,CRITICAL centos:7 #HIGH and CRITICAL

我考试题目中是有两个镜像 centos的镜像不符合要求把对应pod删除即可

https://kubernetes.io/zh/docs/reference/kubectl/cheatsheet/#%E6%A0%BC%E5%BC%8F%E5%8C%96%E8%BE%93%E5%87%BA

5. kube-bench，修复不安全项

这个题目也比较稳就是根据bue-bench对master节点进行修复，基本就是老一套 。根据题目中提示基本没有什么问题！

kube-apiserver

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

- --authorization-mode=Node,RBAC

kubelet

vim /var/lib/kubelet/config.yaml

authentication:

anonymous:

enabled: false

webhook:

enabled: true

authorization:

mode: Webhook

protectKernelDefaults: true

systemctl restart kubelet.service

systemctl status kubelet.service

etcd

mv /etc/kubernetes/manifests/etcd.yaml /etc/kubernetes/

vim /etc/kubernetes/etcd.yaml

- --client-cert-auth=true

1

2

3

这个题也很刺激 。我进入master节点发现kuberentes环境竟然没有启动…what.我修改了配置文件后还是没有起来…中间我的vpn断网了。然后检查的时候环境竟然好了。莫名其妙的…

6. clusterrole

这里考了两次也都会细微的改变。但是无非是以下三个步骤

修改namespace下role的权限只允许对某一类对象做list list的操作

新建一个serviceaccount

创建名为role-2的role，并且通过rolebinding绑定sa-dev-1，只允许对persistentvolumeclaims做update操作。

# kubectl -n db get role

NAME ROLE AGE

sa-dev-1 Role/role-1 7d16h

# kubectl -n db edit role role-1

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

name: role-1

namespace: db

selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/db/roles/role-1

rules:

- apiGroups:

- ""

resources:

- endpoints #只允许对endpoints资源list

verbs:

- list

kubectl -n db create role role-2 --resource=persistentvolumeclaims --verb=update

kubectl create rolebinding role-2-binding --role=role-2 --serviceaccount=db:service-account-web -n db

7. serviceAccount

在qa命名空间内创建ServiceAccount frontend-sa，不允许访问任何secrets。创建名为frontend-sa的Pod使用该ServiceAccount。并删除qa命名空间下没有使用的sa

参照：https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/

apiVersion: v1

kind: ServiceAccount

metadata:

name: frontend-sa

namespace： qa

automountServiceAccountToken: false

...

apiVersion: v1

kind: Pod

metadata:

name: my-pod

namespace: qa

spec:

serviceAccountName: frontend-sa

automountServiceAccountToken: false

...

kubectl get sa -n qa 应该是有三个sa frontend-sa default 还有另外一个 正常应该是保留frontend-sa 删除另外两个？但是default有必要删除吗？哈哈哈哈这个地方有点徘徊！我貌似

8. Dockerfile 和 Pod yaml 检测

这个地方也有点郁闷啊。不是每个文件中让修改两处吗？Dockerfile中我是不是要修改三处呢？题目中注明了 基础镜像是ubuntu：16.04 但是给的文件中是latest应该是要修改的吧？

Dockerfile:

去掉两处 USER root

设置基础镜像为 ubuntu:16.04

Pod yaml:

注释掉 privileged 那一行的相关配置

https://docs.docker.com/develop/develop-images/dockerfile\_best-practices/#from

9. secret

istio-system 命名空间中有一个名称为 db1-test 的 secret， 按照要求完成如下内容：

存储 username 字段到 /home/candidate/user.txt 文件 , password 字段到 /home/candidate/old\_pass.txt 文件。用户需要自己创建文件。

创建secret from user password。

将secret挂载到pod中。

在 istio-system 命名空间创建名称为 db2-test 的 secret，其包含如下内容:

pod name secret-pod

namespace istio-system

container name dev-container

image nginx ###也可能是httpd我考的几次都碰到过

volume name secret-volume

mount path /etc/secret

1. 将secrets base 64写入文件

很多解题都用了kubectl jsonpath的方式个人比较笨就用了传统的方式：

kubectl get secrets -n istio-system db1-test -o yaml

echo "" |base64 -d > /home/candidate/old\_pass.txt

echo "" |base64 -d > /home/candidate/user.txt

注意顺序 查看secrets的时候貌似password都在上面，个人习惯总是容易先搞username。还是容易搞混的

2. 生成secret

参照：https://kubernetes.io/docs/tasks/configmap-secret/managing-secret-using-kubectl/

kubectl create secret generic db2-test -n istio-system \

--from-literal=username=production-instance \

--from-literal=password=KvLftKgs4aHV

3 . secret 挂载到pod中

apiVersion: v1

kind: Pod

metadata:

name: secret-pod

namespace: istio-system

spec:

containers:

- name: dev-container

image: nginx ###我的是httpd

volumeMounts:

- name: secret-volume

mountPath: "/etc/secret"

readOnly: true

volumes:

- name: secret-volume

secret:

secretName: db2-test

10. PodSecurityPolicy

基本就是参照官方：

创建名为 restrict-policy的PodSecurityPolicy，阻止创建privileged Pod

创建名为 restrict-access-role 的ClusterRole允许使用新创建的名为 restrict-policy 的PodSecurityPolicy。

在staging命名空间中创建名为 psp-denial-sa 的serviceAccount。

创建名为deny-access-bind的clusterRoleBinding，绑定刚刚创建的serviceAccount和ClusterRole。

vim restrict-policy.yml

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

metadata:

name: restrict-policy

spec:

privileged: false ###一定要记得这里false吧？

runAsUser:

rule: "RunAsAny"

fsGroup:

rule: "RunAsAny"

seLinux:

rule: "RunAsAny"

supplementalGroups:

rule: "RunAsAny"

kubectl apply -f restrict-policy.yml

kubectl create clusterrole restrict-access-role --verb=use --resource=psp --resource-name=restrict-policy

kubectl create sa psp-denial-sa -n staging

kubectl create clusterrolebinding deny-access-bind --clusterrole=restrict-access-role --serviceaccount=staging:psp-denial-sa

注:参考https://kubernetes.io/docs/concepts/policy/pod-security-policy/#volumes-and-file-systems

11. Container(容器)安全 ,删除包含Volumes或privileged 的pod

查看production namespace下所有pod是否有特权 Privileged 或者挂载 volume 的 pod

kubectl get pods NAME -n production -o jsonpath={.spec.volumes} | jq

kubectl get pods NAME -o yaml -n production | grep "privi.\*: true"

然后删除 Privileged 或者挂载 volume 的 Pod。应该是有3个pod 要删除两个？

12. audit–日志审计

日志审计这个地方很刺激。不知道为什么我改了前两次都没有起来…

现在回想一下我默认的人家写好的都删了这应该是不对的:

\*\* basic policy is provided at /etc/kubernetes/logpolicy’sample-policy.yaml . It onlyspecifies what not to log----指定不记录到日志中的内容\*\*

namespaces changes at RequestResponse level

the request body of persistentwolumes changes inthe namespace front-apps

ConfigMap and Secret changes in all namespaces atthe Metadata level

cat /etc/kubernetes/logpolicy/sample-policy.yaml

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

kind: Policy

omitStages:

- "RequestReceived"

rules:

# 保留指定不记录到日志中的内容（原配置文件内容）

- level: RequestResponse

resources:

- group: ""

resources: ["namespaces"]

- level: Request

resources:

- group: ""

resources: ["persistentvolumes"]

namespaces: ["front-apps"]

- level: Metadata

resources:

- group: "" # core API group

resources: ["secrets", "configmaps"]

- level: Metadata

omitStages:

- "RequestReceived"

kube-apiserver.yaml

配置文件路径，log路径 最大保留天数，保留的审计日志文件的最大数量

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

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

- --audit-log-path=/var/log/kubernetes/audit.log

- --audit-log-maxage=10

- --audit-log-maxbackup=1

重启服务并验证

systemctl restart kubelet

kubectl apply -f xxx.yaml

tail -f /var/log/kubernetes/audit.log

注： 修改之前先做好备份！谨记参照官方：https://kubernetes.io/docs/tasks/debug-application-cluster/audit/。文件的挂载环境中已配置好了不用修改添加的。

13. image policy

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

admission-control.conf or|dmission-control.yaml

vim /etc/kubernetes/admission-control/admission-control.conf

apiVersion: apiserver.config.k8s.io/v1

kind: AdmissionConfiguration

plugins:

- name: ImagePolicyWebhook

path: imagepolicy.conf

1

2

3

4

5

6

imagepolicy.conf | imagepolicy.json

vim /etc/kubernetes/admission-control/imagepolicy.conf

{

"imagePolicy": {

"kubeConfigFile": "/etc/kubernetes/admission-control/imagepolicy\_backend.kubeconfig",

"allowTTL": 50,

"denyTTL": 50,

"retryBackoff": 500,

"defaultAllow": false

}

}

注： 只修改了defaultAllow true为false!

1

2

3

4

5

6

7

8

9

10

11

imagepolicy\_backend.kubeconfig

vim /etc/kubernetes/admission-control/imagepolicy\_backend.kubeconfig

apiVersion: v1

kind: Config

clusters:

- name: trivy-k8s-webhook

cluster:

certificate-authority: /etc/kubernetes/admission-control/imagepolicywebhook-ca.crt

server: https://acg.trivy.k8s.webhook:8090/scan

contexts:

- name: trivy-k8s-webhook

context:

cluster: trivy-k8s-webhook

user: api-server

current-context: trivy-k8s-webhook

preferences: {}

users:

- name: api-server

user:

client-certificate: /etc/kubernetes/admission-control/api-server-client.crt

client-key: /etc/kubernetes/admission-control/api-server-client.key

#只在server中增加了相关配置！

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

kube-apiserver

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

- --admission-control-config-file=/etc/kubernetes/admission-control/admission-control.conf

- --enable-admission-plugins=NodeRestriction,ImagePolicyWebhook

1

2

3

验证

systemctl restart kubelet

kubectl apply -f /root/xxx/vulnerable-manifest.yaml

tail -n 10 /var/log/imagepolicy/roadrunner.log

1

2

3

14. api-server 参数调整

这个题没有读懂。没有搞明白 只看懂了apiserver中修改了：

- --enable-admission-plugins=AlwaysAdmit

修改为：

- --enable-admission-plugins=NodeRestriction

1

2

3

然后根据题目提示还删除了了一个匿名的clusterrole…接下来就不知道要怎么搞了。正常就是这个题得分应该是没有的

15. AppArmor

注： 参照https://kubernetes.io/docs/tutorials/clusters/apparmor/

ssh到work节点

cat /etc/apparmor.d/nginx\_apparmor

#include <tunables/global>

profile nginx-profile-3 flags=(attach\_disconnected) {

#include <abstractions/base>

file,

# Deny all file writes.

deny /\*\* w,

}

sudo apparmor\_status | grep nginx

sudo -qa /etc/apparmor.d/nginx\_apparmor

sudo apparmor\_status | grep nginx

nginx-profile-3

1

2

3

4

5

6

7

8

9

10

11

12

13

14

exit work节点回到跳板机：

vim https://github.com/lmtbelmonte/cks-cert

apiVersion: v1

kind: Pod

metadata:

name: writedeny

namespace: dev

annotations:

container.apparmor.security.beta.kubernetes.io/busybox: localhost/nginx-profile-3

spec:

containers:

- name: busybox

image: busybox:1.33.1

1

2

3

4

5

6

7

8

9

10

11

12

kubecl apply -f ~/nginx-deploy.yml

1

其他：

————————————————

版权声明：本文为CSDN博主「saynaihe」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/saynaihe/article/details/122133395