**CKS Exam Questions and Answers**

1. Mirror scan ImagePolicyWebhook  
2. sysdig detects pods  
3. clusterrole  
4. AppArmor  
5. PodSecurityPolicy  
6. Network Policy  
7. dockerfile detection and yaml file problem  
8. pod security  
9. Create SA  
10. trivy detects mirror security  
11. Create secret  
12. kube-bench  
13. gVsior  
14. Audit  
15. Default Network Policy  
16. Falco detection output log format

kubernetes exam in action  
———————————————  
Copyright statement: This article is an original article by CSDN blogger "ghostwritten" and follows the CC 4.0 BY-SA copyright agreement. Please attach the original source link and this statement for reprinting.  
Original link: <https://blog.csdn.net/xixihahalelehehe/article/details/122525427>

Exam Information  
2 hours  
15-20 Topics  
The appointment time is the same as CKA, and the results will be released within 32 hours  
Out of 100, 87 or 93, but a passing score of 67  
Simulation environment  
4 environments 1 console  
NAT network segment 192.168.26.0  
Mock exam questions

**1. Mirror scan ImagePolicyWebhook**

switch cluster kubectl config use-context k8s  
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’s master node, where all services and files have been prepared and placed  
Glven an incomplete configuration in directory /etc/kubernetes/aa and a functional container image scanner with HTTPS sendpitont <http://192.168.26.60:1323/image_policy>  
  
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 resource /csk/1/web1.yaml  
Problem solving ideas  
ImagePolicyWebhook  
  
Keywords: image\_policy, deny  
1. Switch the cluster, view the master, sshmaster  
2. ls /etc/kubernetes/xxx  
3. vi /etc/kubernetes/xxx/xxx.yaml change true to false  
The address of https in vi /etc/kubernetes/xxx/xxx.yaml  
volume needs to be mounted  
4. Enable ImagePolicyWebhook and --admission-control-config-file=  
5. systemctl restart kubelet  
6. kubectl run pod1 --image=nginx  
  
**Case:**  
  
configure /etc/kubernetes/manifests/kube-apiserver.yaml  
Add ImagePolicyWebhook related policies  
Restart api-server,systemctl restart kubelet  
Failed to verify image creation pod  
Modify /etc/kubernetes/admission/admission\_config.yaml policy defaultAllow: true  
Revalidate the image to create the pod  
$ ls /etc/kubernetes/aa/  
admission\_config.yaml apiserver-client-cert.pem apiserver-client-key.pem external-cert.pem external-key.pem kubeconf  
1  
2  
$ cd /etc/kubernetes/aa  
$ cat kubeconf  
apiVersion: v1  
kind: Config  
  
# clusters refers to the remote service.  
clusters:  
- cluster:  
    certificate-authority: /etc/kubernetes/aa/external-cert.pem # CA for verifying the remote service.  
    server: <http://192.168.26.60:1323/image_policy> # URL of remote service to query. Must use 'https'.  
  name: image-checker  
  
contexts:  
- context:  
    cluster: image-checker  
    user: api-server  
  name: image-checker  
current-context: image-checker  
preferences: {}  
  
# users refers to the API server's webhook configuration.  
users:  
- name: api-server  
  user:  
    client-certificate: /etc/kubernetes/aa/apiserver-client-cert.pem # cert for the webhook admission controller to use  
    client-key: /etc/kubernetes/aa/apiserver-client-key.pem # key matching the cert  
  
  
$ cat admission\_config.yaml  
apiVersion: [apiserver.config.k8s.io/v1](http://apiserver.config.k8s.io/v1)  
kind: AdmissionConfiguration  
plugins:  
  - name: ImagePolicyWebhook  
    configuration:  
      imagePolicy:  
        kubeConfigFile: /etc/kubernetes/aa/kubeconf  
        allowTTL: 50  
        denyTTL: 50  
        retryBackoff: 500  
        defaultAllow: false  
  
#Modify api-server configuration  
$ cat /etc/kubernetes/manifests/kube-apiserver.yaml  
..........................................  
  - command:  
    - kube-apiserver  
    - --admission-control-config-file=/etc/kubernetes/aa/admission\_config.yaml #Add this line  
    - --advertise-address=192.168.211.40  
    - --allow-privileged=true  
    - --authorization-mode=Node,RBAC  
    - --client-ca-file=/etc/kubernetes/pki/ca.crt  
    - --enable-admission-plugins=NodeRestriction,ImagePolicyWebhook # #Modify this line  
    - --enable-bootstrap-token-auth=true  
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt  
  ............  
    - mountPath: /etc/kubernetes/pki  
      name: k8s-certs  
      readOnly: true  
    - mountPath: /etc/kubernetes/aa #Add this line  
      name: k8s-admission #Add this line  
      readOnly: true #Add this line  
............  
  - hostPath: #Add this line  
      path: /etc/kubernetes/aa #Add this line  
      type: DirectoryOrCreate #Add this line  
    name: k8s-admission #Add this line  
  - hostPath:  
      path: /usr/local/share/ca-certificates  
      type: DirectoryOrCreate  
    name: usr-local-share-ca-certificates  
  - hostPath:  
      path: /usr/share/ca-certificates  
      type: DirectoryOrCreate  
    name: usr-share-ca-certificates  
status: {}  
  
  
$ k get nodes  
NAME STATUS ROLES AGE VERSION  
master Ready control-plane, master 9d v1.20.1  
node1 Ready <none> 9d v1.20.1  
node2 Ready <none> 9d v1.20.1  
  
# Failed to create pod  
$ k run test --image=nginx  
Error from server (Forbidden): pods "test" is forbidden: Post "<https://external-service:1234/check-image?timeout=30s>": dial tcp: lookup external-service on [8.8.8.8:53](http://8.8.8.8:53/): no such host  
  
#Modify admission\_config.yaml configuration  
$ vim /etc/kubernetes/aa/admission\_config.yaml  
apiVersion: [apiserver.config.k8s.io/v1](http://apiserver.config.k8s.io/v1)  
kind: AdmissionConfiguration  
plugins:  
  - name: ImagePolicyWebhook  
    configuration:  
      imagePolicy:  
        kubeConfigFile: /etc/kubernetes/aa/kubeconf  
        allowTTL: 50  
        denyTTL: 50  
        retryBackoff: 500  
        defaultAllow: true #Modify this behavior true  
  
  
#Restart api-server  
$ ps -ef | grep api  
root 78871 39023 0 20:17 pts/3 00:00:00 grep --color=auto api  
  
$ mv ../kube-apiserver.yaml .  
  
#Create pod successfully  
$ k run test --image=nginx  
pod/test created  
  
  
**2. sysdig detects pods**

switch cluster kubectl config use-context k8s  
you may user you brower to open one additonal tab to access sysdig’s documentation ro Falco’s 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 available to use:  
  
sysdig  
falico  
the tools are pre-installed on the cluster’s worker node only; the are not available on the base system or the master node.  
using the tool of your choice (including any non pre-install tool) analyse the container's 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]  
1  
Problem solving ideas  
Sysdig User Guide  
  
keyword: sysdig  
0. Remember to use sysdig -l |grep to search for relevant fields  
1. Switch the cluster, query the corresponding pod, and ssh to the node host corresponding to the pod  
2. Use sysdig, pay attention to the required format and time, and redirect the result to the corresponding file  
3. sysdig -M 30 -p "\*%evt.time,%user.uid,%[proc.name](http://proc.name/)" [container.id](http://container.id/)=container id >/opt/2/report  
  
case  
  
**3. clusterrole**

switch cluster kubectl config use-context k8s  
context  
A Role bound to a pod’s serviceAccount grants overly permissive permission  
Complete the following tasks to reduce the set of permissions.  
task  
Glven an existing Pod name web-pod running in the namespace monitoring Edit the Roleebound 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 namespaces monitoring which only allows performing update operations, only on resources of type persistentvoumeclaims.  
create a new Rolebind name role role-2-bindding binding the newly created Roleto the Pod’s serviceAccount  
  
Problem solving ideas  
RBAC  
  
Keywords: role, rolebinding  
1. Find the rolle modification permissions corresponding to rollebind as list and endpoints  
$ kubectl edit role role-1 -n monitoring  
2. Remember --verb is permission --resource is object  
$ kubectl create role role-2 --verb=update --resource=persistentvolumeclaims -n  
monitoring  
3. Create a binding and bind it to the corresponding sa  
$ kubectl create rolebinding role-2-bindding --role=role-2 --  
serviceaccount=monitoring:sa-dev-1 -n monitoring  
  
**4. AppArmor**

switch cluster kubectl config use-context k8s  
Context  
AppArmor is enabled on the cluster’s 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's 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  
  
Problem solving ideas  
apparmor  
  
keyword: apparmor  
1. Switch the cluster, remember to check the nodes, ssh to the node node  
2. View the corresponding configuration file and name  
$ cd /etc/apparmor.d  
$ vi nginx\_apparmor  
$ apparmor\_status |grep nginx-profile-3 # There is no grep to indicate that it is not started  
$ apparmor\_parser -q nginx\_apparmor # Load and enable this configuration file  
3. Modify the corresponding yaml to apply this rule, open the URL copy example of the official website, modify the container name and the local configuration name  
$ vi /cks/4/pod1.yaml  
apiVersion: v1  
kind: Pod  
metadata:  
  name: hello-apparmor  
  annotations:  
    [container.apparmor.security.beta.kubernetes.io/hello:localhost/nginx-profile-3](http://container.apparmor.security.beta.kubernetes.io/hello:localhost/nginx-profile-3)  
spec:  
  containers:  
  - name: hello  
    image: busybox  
    command: [ "sh", "-c", "echo 'Hello AppArmor!' && sleep 1h" ]  
  
4. Create after modification  
$ kubectl apply -f /cks/4/pod1.yaml  
  
**5. PodSecurityPolicy**

switch cluster kubectl config use-context k8s63  
context  
A PodsecurityPolicy shall prevent the create 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 pspdenial-sa in the existing namespace development .  
Finally, create a new clusterRoleBinding named dany-access-bind , which binds the newly created ClusterRole restrict-access-role to the newly created serviceAccount  
Problem solving ideas  
PodSecurityPolicy  
  
Keywords: psp policy privileged  
0. Switch the group to see if it is enabled  
$ vi /etc/kubernetes/manifests/kube-apiserver.yaml  
- --enable-admission-plugins=NodeRestriction,PodSecurityPolicy  
  
$ systemctl restart kubelet  
  
1. Copy the psp from the official website, modify the deny privilege  
$ cat psp.yaml  
apiVersion: policy/v1beta1  
kind: PodSecurityPolicy  
metadata:  
name: prevent-psp-policy  
spec:  
privileged: false  
seLinux:  
rule: RunAsAny  
supplementalGroups:  
rule: RunAsAny  
runAsUser:  
rule: RunAsAny  
fsGroup:  
rule: RunAsAny  
volumes:  
- '\*'  
  
$ kubectl create -f psp.yaml  
  
2. Create the corresponding clusterrole  
$ kubectl create clusterrole restrict-access-role --verb=use --resource=podsecuritypolicy --resource-name=prevent-psp-policy  
  
3. Create sa to see the corresponding ns  
$ kubectl create sa psp-denial-sa -n development  
  
4. Create a binding relationship  
$ kubectl create clusterrolebinding dany-access-bind --clusterrole=restrict-access-role --serviceaccount=development:psp-denial-sa  
  
**6. Network Policy**

switch cluster kubectl config use-context k8s  
create a NetworkPolicy named pod-access to restrict access to Pod products-service running in namespace development . 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  
Problem solving ideas  
NetworkPolicy  
  
Keywords: NetworkPolicy  
1. The host checks the label of the pod  
$ kubectl get pod -n development --show-labels  
  
2. Check the label corresponding to ns, there is no need to set it  
$ kubectl label ns testing name=testing  
  
3. Orchestrate network policy  
$ cat /cks/6/p1.yaml  
kind: NetworkPolicy  
metadata:  
name: "pod-access"  
namespace: "development"  
spec:  
  podSelector:  
    matchLabels:  
      environment: staging  
  policyTypes:  
  -Ingress  
  ingress:  
  - from:  
    - namespaceSelector:  
        matchLabels:  
          name: testing  
  - from:  
    - namespaceSelector:  
        matchLabels:  
      podSelector:  
        matchLabels:  
          environment: staging  
           
  
$ kubectl create -f /cks/6/p1.yaml  
  
**7. dockerfile detection and yaml file problem**

switch cluster kubectl config use-context k8s  
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.  
  
Problem solving ideas  
  
Keywords: Dockerfile issues  
1. Pay attention to the number of errors prompted by the dockerfile  
Note: USER root  
2.yaml problem: pay attention to the api version problem, and the privileged network and mirror version, also depends on the errors mentioned in the title  
  
Case:  
Dockerfile  
  
# build container stage 1  
FROM ubuntu:20.04  
ARG DEBIAN\_FRONTEND=noninteractive  
RUN apt-get update && apt-get install -y golang-go=2:1.13~1ubuntu2  
COPY app.go .  
RUN pwd  
RUN CGO\_ENABLED=0 go build app.go  
  
# app container stage 2  
FROM alpine:3.12.0  
RUN addgroup -S appgroup && adduser -S appuser -G appgroup -h /home/appuser  
RUN rm -rf /bin/\*  
COPY --from=0 /app /home/appuser/  
USER appuser  
cmd ["/home/appuser/app"]  
  
**8. pod security**

8. pod security  
switch cluster kubectl config use-context k8s  
context  
lt 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.  
  
Problem solving ideas  
  
Keywords: stateless immutable  
1. get all pods  
2. Check if there is a privilege privi\*  
3. Check if there is volume  
4. Delete the privileged network and volume  
$ kubectl get pod pod1 -n testing -o jsonpath={.spec.volumes} | jq  
$ kubectl get pod sso -n testing -o yaml |grep "privi.\*: true"  
$ kubectl delete pod xxxxx -n testing  
  
**9. Create SA**

switch cluster kubectl config use-context k8s  
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.lnspect the Pod named frontend running in the namespace qa . Edit the Pod to use the newly created serviceAccount  
  
Problem solving ideas  
Configure Service Accounts for Pods  
  
Keyword: ServiceAccount "must not have access to any secrets"  
1. Get the sa template  
$ kubectl create serviceaccount frontend-sa -n qa --dry-run -o yaml  
2. Find automatic mounting through official documentation  
$ k edit pod frontend -n qa  
apiVersion: v1  
kind: Pod  
metadata:  
  creationTimestamp: null  
  labels:  
    run: frontend  
  name: frontend  
spec:  
  serviceAccountName: frontend-sa #Add this line  
  automountServiceAccountToken: false #Add this line  
  containers:  
  - image: nginx  
    name: frontend  
    resources: {}  
  dnsPolicy: ClusterFirst  
  restartPolicy: Always  
status: {}  
  
3. Modify the serviceAccountName in the pod  
4. Create pod to delete other sa  
  
**10. Trivy detects mirror security**  
switch cluster kubectl config use-context k8s  
**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  
  
**Problem solving ideas**  
  
Keywords: Trivy scanner High or Critical  
1. Switch the cluster and ssh to the corresponding master  
2. get pod scans the corresponding images, no High or Critical  
$ docker run [ghcr.io/aquasecurity/trivy:latest](http://ghcr.io/aquasecurity/trivy:latest) image nginx:latest |grep 'High|Critical'  
3. Delete the problematic mirror pod  
$ docker rmi <image>  
  
**11. Create Secret**

switch cluster kubectl config use-context k8s  
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't exist yet. Do not use/modify the created files in the following steps, create new temporary files 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 avolume:  
  
pod name dev-pod  
namespace istio-system  
container name dev-container  
image nginx:1.9  
volume name dev-volume  
mount path /etc/test-secret  
Problem solving ideas  
Secret  
  
keyword: secret  
1. Obtain the username and passwd of db1-test  
$ kubectl get secrets db1-test -n istio-system -o yaml  
$ echo -n "aGFoYTAwMQ==" | base64 -d > /cks/11/old-pass.txt  
$ echo -n "dG9t" | base64 -d > /cks/11/old-username.txt  
  
2. Create a secret named test-workflow  
$ kubectl create secret generic test-workflow --from-literal=username=thanos --from-literal=password=hahahaha -n istio-system  
  
3. More requirements to create pods of secrets  
$ cat secret-pod.yaml  
apiVersion: v1  
kind: Pod  
metadata:  
  name: dev-pod  
  namespace: istio-system  
spec:  
  containers:  
  - name: dev-container  
    image: nginx:1.9  
    volumeMounts:  
    - name: foo  
      mountPath: "/etc/test-secret"  
      readOnly: true  
  volumes:  
  - name: dev-volume  
    secret:  
      secretName: test-workflow  
  
k create -f secret-pod.yaml  
  
**12. kube-bench**

switch cluster kubectl config use-context k8s65  
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 theaffected 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  
  
Problem solving ideas  
  
Keywords: look at the entry to determine whether it is a scan  
1. Switch the machine to the corresponding ssh to master node  
2. kube-bench run Find the corresponding entry, and then fix it  
docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -t aquasec/kube-bench:latest master --version 1.20  
There is an ETCD in the exam  
  
Case 1  
  
$ docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -t aquasec/kube-bench:latest master --version 1.20  
......  
[FAIL] 1.1.12 Ensure that the etcd data directory ownership is set to etcd:etcd (Automated)  
.............

$ cat /etc/kubernetes/kubelet.conf  
apiVersion: v1  
clusters:  
- cluster:  
    certificate-authority-data:   
    server: [https://192.168.211.40:6443](https://192.168.211.40:6443/)  
  name: kubernetes  
contexts:  
- context:  
    cluster: kubernetes  
    user: system:node:master  
  name: system:node:master@kubernetes  
current-context: system:node:master@kubernetes  
kind: Config  
preferences: {}  
users:  
- name: system:node:master  
  user:  
    client-certificate: /var/lib/kubelet/pki/kubelet-client-current.pem  
    client-key: /var/lib/kubelet/pki/kubelet-client-current.pem  
  
  
$ echo  | base64 -d > /etc/kubernetes/pki/apiserver-kubelet-ca.crt  
  
$ cat /etc/kubernetes/pki/apiserver-kubelet-ca.crt  
-----BEGIN CERTIFICATE-----  
MIIC5zCCAc+gAwIBAgIBADANBgkqhkiG9w0BAQsFADAVMRMwEQYDVQQDEwprdWJl  
cm5ldGVzMB4XDTIxMDkxMzA4MTcwOVoXDTMxMDkxMTA4MTcwOVowFTETMBEGA1UE  
AxMKa3ViZXJuZXRlczCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOPA  
wydYcbRAZ7F2jQjjGIqVfPSdyUxS1LB0e0Jhgtb1rUpmv12uCmfUPhIDRvzvKhfs  
yr06avNnsfIvRzr+zj1zCOX3TSZbf4kCZ8N88JRIDt6pCKIISLe8uksuJO095ejv  
nVuoGmBFeKlcuz1GKQKTL7jSZ7+4MrMayE9HdnbzuZMTN6fSoExF1xq3oC0i+eBB  
3T+Pc1yyWCMrwWXG9UFf4Z8xXDhgn/hDJHJERvyklFjqxjZD+xPyKq88vO9K+JmV  
ly6Ll5kmel7OtOYMDg1i0S0IQfI0kTiOvwCl8s55PWQ8mNkYryIYxKNBAO/x/QB8  
00Yk4afVDIBYWduHr1kCAwEAAaNCMEAwDgYDVR0PAQH/BAQDAgKkMA8GA1UdEwEB  
/wQFMAMBAf8wHQYDVR0OBBYEFG9NaDMQIT4ss05ZSLg/WDbl6zdiMA0GCSqGSIb3  
DQEBCwUAA4IBAQCikJPGwsAwaVxqlWk2oGunnG77P4eqZbYtwZLnhoviKGSubX5r  
r0skRWGcTT12PCLiEc2hLvljvT/lQ3/Muv6nbYDPSx9b3EoEF2YxHqWMc5BXJU/9  
NpAXc+m/7MrZAepW1sw+ncSTdH01NbQ1BBz82kFzMYNoJ+LJcExlvSkzrMuWCWiA  
6vbljHFrJBw5kE1Ow/GNK9HTTR99zuoe8LrRojsPATf/vzAJdLQki71riYtsFE04  
YOreqA8cfMxdnPcAxo9gRVC0aC0DwaQx/Z5+cE70UmyvqPrn8Tbz1Mn9kuWePMa9  
OW4b7Stb9Zz6PB7zjK97txsxtEEeu+/Lx2W+  
-----END CERTIFICATE-----  
  
  
$ vim  /etc/kubernetes/manifests/kube-apiserver.yaml  
.....  
--kubelet-certificate-authority=/etc/kubernetes/pki/apiserver-kubelet-ca.crt  
...................  
  
$ kubectl get pods -n kube-system | grep kube-apiserver  
$ docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -t aquasec/kube-bench:latest master --version 1.20  
  
**13. gVsior**

Change the cluster kubectl config use-context k8s67  
context  
This cluster uses containerd as CRl runtime. Containerd’s 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 named runsc . 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  
  
Problem solving ideas  
RuntimeClass  
  
Keywords: gVisor  
1. Switch the cluster and create a runtimeclass with the official website documentation  
$ vim rc.yaml  
apiVersion: [node.k8s.io/v1beta1](http://node.k8s.io/v1beta1)  
kind: RuntimeClass  
metadata:  
  name: untrusted  
handler: runsc  
  
$ k -f rc.yaml create  
  
2. Another topic requires creating a pod to use this runtime  
$ k edit pod mypod -n client  
apiVersion: v1  
kind: Pod  
metadata:  
  name: mypod  
  namespace: client  
spec:  
  runtimeClassName: untrusted  
......  
  
**14. Audit**

switch cluster kubectl config use-context k8s  
Task  
Enable audit logs in the cluster. To do so, enable the log backend, and ensure that:  
  
logs are stored at /var/log/kubernetes/audit-logs.txt  
log files are retained for 5 days 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 the cluster’s master node. Edit and extend the basic policy to log:  
  
namespaces changes at RequestResponse level  
the request body of pods changes in the namespace front-apps  
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. Don’t forget to apply  
Problem solving ideas  
audit  
  
keyword: policy  
1. Switch the cluster to log in to the master, then create a directory, modify yaml, and enable auditing  
$ mkdir /var/log/kubernetes/  
$ mkdir /etc/kubernetes/logpolicy/  
$ cat /etc/kubernetes/logpolicy/sample-policy.yaml  
$ cat policy.yaml  
apiVersion: [audit.k8s.io/v1](http://audit.k8s.io/v1)  
kind: Policy  
omitStages:  
  - "RequestReceived"  
rules:  
  - level: RequestResponse  
    resources:  
    - group: ""  
      resources: ["namespaces"]  
  
  - level: Request  
    resources:  
    - group: "" # core API group  
      resources: ["pods"]  
    namespaces: ["front-apps"]  
  
  - level: Metadata  
    resources:  
    - group: ""  
      resources: ["secrets","configmaps"]  
  
  - level: Metadata  
    omitStages:  
      - "RequestReceived"  
  
2. More official website documents to modify the corresponding strategy  
$ vim /etc/kubernetes/manifests/kube-apiserver.yaml  
    - --audit-policy-file=/etc/kubernetes/logpolicy/sample-policy.yaml # add  
    - --audit-log-path=/var/log/kubernetes/audit-logs.txt # add  
    - --audit-log-maxage=5 # add  
    - --audit-log-maxbackup=10  
......  
   - mountPath: /etc/kubernetes/logpolicy # add  
      name: audit # add  
  hostNetwork: true  
  priorityClassName: system-node-critical  
  volumes:  
  - hostPath: # add  
      path: /etc/kubernetes/logpolicy # add  
      type: DirectoryOrCreate # add  
    name: audit # add  
  
  
3. Restart kubelet  
$ systemctl restart kubelet  
$ k get pods -n kube-system | grep api  
$ cat /var/log/kubernetes/audit-logs.txt

**15. Default Network Policy**

switch cluster kubectl config use-context k8s  
context  
A default-deny NetworkPolicy avoids to accident all y 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 lngress  
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  
  
Problem solving ideas  
NetworkPolicy  
  
Keywords: NetworkPolicy defined  
1. Observe clearly whether all conditions are rejected by default or other conditions, and more topics require official documents to write yaml  
$ cat denynetwork.yaml  
apiVersion: [networking.k8s.io/v1](http://networking.k8s.io/v1)  
kind: NetworkPolicy  
metadata:  
  name: denynetwork  
  namespace: development  
spec:  
  podSelector: {}  
  policyTypes:  
  -Ingress  
  
$ k create -f denynetwork.yaml  
  
**16. Falco detection output log format**  
  
  
$ ssh node1  
$ systemctl stop falco  
$ falco  
$ cd /etc/falco/  
$ ls  
falco\_rules.local.yaml falco\_rules.yaml falco.yaml k8s\_audit\_rules.yaml rules.available rules.d  
  
$ rep -r "A shell was spawned in a container with an attached terminal" \*  
falco\_rules.yaml: A shell was spawned in a container with an attached terminal (user=%[user.name](http://user.name/) user\_loginuid=%user.loginuid %[container.info](http://container.info/)  
  
  
#update configuration  
root@node1:/etc/falco# cat falco\_rules.local.yaml  
- rule: Terminal shell in container  
  desc: A shell was used as the entrypoint/exec point into a container with an attached terminal.  
  condition: >  
    spawned\_process and container  
    and shell\_procs and proc.tty != 0  
    and container\_entrypoint  
    and not user\_expected\_terminal\_shell\_in\_container\_conditions  
  output: >  
    %evt.time,%[user.name](http://user.name/),%[container.name](http://container.name/),%[container.id](http://container.id/)  
    shell=%[proc.name](http://proc.name/) parent=%proc.pname cmdline=%proc.cmdline terminal=%proc.tty container\_id=%[container.id](http://container.id/) image=%container.image.repository)  
  priority: WARNING  
  tags: [container, shell, mitre\_execution]  
  
$ falco  
Mon May 24 00:07:13 2021: Falco version 0.28.1 (driver version 5c0b863ddade7a45568c0ac97d037422c9efb750)  
Mon May 24 00:07:13 2021: Falco initialized with configuration file /etc/falco/falco.yaml  
Mon May 24 00:07:13 2021: Loading rules from file /etc/falco/falco\_rules.yaml:  
Mon May 24 00:07:13 2021: Loading rules from file /etc/falco/falco\_rules.local.yaml: #Configuration takes effect  
Mon May 24 00:07:13 2021: Loading rules from file /etc/falco/k8s\_audit\_rules.yaml:  
Mon May 24 00:07:14 2021: Starting internal webserver, listening on port 8765  
00:07:30.297671117: Warning Shell history had been deleted or renamed (user=root user\_loginuid=-1 type=openat command=bash [fd.name](http://fd.name/)=/root/.bash\_history name=/root/.bash\_history path=<NA> oldpath =<NA> k8s\_apache\_apache\_default\_3ece2efb-fe49-4111-899f-10d38a61bab6\_0 (id=84dd6fe8a9ad))  
  
format change  
00:07:33.763063865: Warning 00:07:33.763063865,root,k8s\_apache\_apache\_default\_3ece2efb-fe49-4111-899f-10d38a61bab6\_0,84dd6fe8a9ad shell=bash parent=runc cmdline=bash terminal=34816 container\_id=84addd6fe8a9)

<https://blog.csdn.net/xixihahalelehehe/article/details/122525427>