# kube-bench 修复不安全项

|  |
| --- |
| kubectl config use-context KSCS00201  ssh master01  sudo -i  # 检查命令  kube-bench master（apiserver）  vim /etc/kubernetes/manifests/kube-apiserver.yaml  修改： - --authorization-mode=**Node,RBAC**  **删除**： - --insecure-bind-address=0.0.0.0  kube-bench node（kubelet）  vim /var/lib/kubelet/config.yaml  anonymous: #修改anonymous下的，将true改为false  enabled: **false** #改为false  authorization: #修改authorization下的  mode: **Webhook** #改为Webhook  #编辑完后重新加载配置文件，并重启kubelet  systemctl daemon-reload  systemctl restart kubelet.service  kube-bench （etcd）  vim /etc/kubernetes/manifests/etcd.yaml  修改： - --client-cert-auth=**true** #修改为true |

# 2、Pod指定ServiceAccount

|  |
| --- |
| kubectl config use-context KSCS00201  1 创建ServiceAccount  cat pod-sa.yaml  apiVersion: v1  kind: ServiceAccount  metadata:  name: heian  namespace: dev  **automountServiceAccountToken**: **false** #（重点）  2、创建Pod使用该ServiceAccount  cat pod-sa-mount.yaml  apiVersion: v1  kind: Pod  metadata:  name: heian  namespace: dev  spec:  **serviceAccountName**: heian # 重点  containers:  - image: nginx  name: heian  resources: {}  3. 删除没有使用的ServiceAccount  查看所有sa  kubectl get sa -n qa  查看已经在用的sa  kubectl get pod -n qa -o yaml | grep -i serviceAccountName  删除不用的sa  kubectl delete sa test01 -n qa |

# 3、默认网络策略

|  |
| --- |
| kubectl config use-context KSCS00201  cat test-networkpolicy.yaml  apiVersion: networking.k8s.io/v1  kind: NetworkPolicy  metadata:  name: default-deny-all  namespace: test  spec:  **podSelector**: {}  policyTypes:  - **Ingress** #注意看题，是Ingress + Egress（入口+出口），还是只是Ingress 或只是Egress。  - **Egress** #在1.23的考试中，只要求拒绝所有Egress流量，那就只写这这个- Egress即可，这种情况就不要写- Ingress 了  kubectl apply -f test-networkpolicy.yaml |

# 4、RBAC - RoleBinding

|  |
| --- |
| kubectl config use-context KSCH00201  kubectl describe rolebindings -n db  kubectl edit role role-1 -n db  rules: #模拟环境里要删除掉null，然后添加以下内容。考试时，要根据实际情况修改。  - apiGroups: [""]  resources: ["services"] #只允许对 services资源类型执行 get 操作  verbs: ["get"]  检查  kubectl describe role role-1 -n db  kubectl create role role-2 --verb=delete --resource=namespaces -n db  kubectl create rolebinding role-2-binding --role=role-2 --serviceaccount=db:service-account-web -n db  检查  kubectl describe rolebindings -n db |

# 5、日志审计 log audit

|  |
| --- |
| kubectl config use-context KSCH00701  (1) 编写日志审计策略文件。  apiVersion: audit.k8s.io/v1  kind: Policy  omitStages:  - "RequestReceived"  rules:  - level: RequestResponse  resources:  - group: ""  resources: ["namespaces"]  - level: Request  resources:  - group: ""  resources: ["persistentvolumes"]  namespaces: ["front-apps"]  - level: Metadata  resources:  - group: ""  resources: ["secrets", "configmaps"]  - level: Metadata  omitStages:  - "RequestReceived"  (2) 修改 kube-apiserver.yaml 配置文件，启用日志审计策略，日志策略配置文件位置、日志文件存储位置、循环周期。  # /etc/kubernetes/manifests/kube-apiserver.yaml  ...  # 设置日志审计策略文件在 pod 里的 mount 位置  - --audit-policy-**file**=/etc/kubernetes/logpolicy/sample-policy.yaml  # 设置日志文件存储位置  - --audit-log-**path**=/var/log/kubernetes/audit-logs.txt  # 设置日志文件循环  - --audit-log-**maxage**=10  - --audit-log-**maxbackup**=2  # mount 日志策略和日志文件的  volumeMounts:  - mountPath: /etc/kubernetes/logpolicy/sample-policy.yaml  name: **audit**  readOnly: true  - mountPath: /var/log/kubernetes/audit-logs.txt  name: **audit-log**  readOnly: false  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  重启 kubelet。 |

# 6、创建 Secret

|  |
| --- |
| kubectl config use-context KSCH00701  1 将db1-test的username和password，通过base64解码保存到题目指定文件：  kubectl get secrets -n istio-system db1-test -o jsonpath={.data.username} | base64 -d > /cks/sec/user.txt  kubectl get secrets -n istio-system db1-test -o jsonpath={.data.password} | base64 -d > /cks/sec/pass.txt  2 创建名为 db2-test 的 secret 使用题目要求的用户名和密码作为键值。注意要加命名空间。  kubectl create secret generic db2-test -n istio-system --from-literal=username=production-instance --from-literal=password=KvLftKgs4aVH  3 根据题目要求，参考官网，创建Pod使用该secret  cat pod-secrets.yaml  apiVersion: v1  kind: Pod  metadata:  name: secret-pod  namespace: test  spec:  containers:  - name: dev-container  image: nginx  volumeMounts:  - name: **secret-volume**  mountPath: "/etc/secret"  volumes:  - name: **secret-volume**  **secret**:  secretName: **db2-test**  kubectl apply -f pod-secrets.yaml |

# 7、Dockerfile检测

|  |
| --- |
| kubectl config use-context KSCH00701  <1> 修改Dockerfile  1、仅将CMD上面的USER root修改为USER nobody，不要改其他的USER root。  **USER nobody**  2、修改基础镜像为题目要求的 ubuntu:16.04  **FROM ubuntu:16.04**  <2> 修改deployment.yaml  1、**template 里标签跟上面的内容不一致**，所以需要将原先的run: couchdb 修改为app: couchdb  2、**删除**  'SYS\_ADMIN' 字段，确保 'privileged': 为False 。可以注释：  securityContext:  {'Capabilities': {'add': ['NET\_BIND\_SERVICE'], 'drop': ['all']}, 'privileged': False, 'readonlyRootFilesystem': False, 'runAsUser': 65535} |

# 8、沙箱运行容器 gVisor

|  |
| --- |
| kubectl config use-context KSMV00301  1 创建RuntimeClass  cat rc.yaml  apiVersion: node.k8s.io/v1beta1  kind: RuntimeClass  metadata:  name: untrusted  namespace: test  **handler: runsc** # 重点  kubectl apply -f rc.yaml  2 将命名空间为server下的Pod引用RuntimeClass。  考试时，3个Deployment下有3个Pod，修改3个deployment即可。    kubectl -n server get deployment  编辑deployment  kubectl -n server edit deployments busybox-run  spec: #找到这个spec，注意在deployment里是有两个单独行的spec的，要找第二个，也就是下面有containers这个字段的spec。  **runtimeClassName: untrusted** #添加这一行，注意空格对齐，保存会报错，忽略即可。  containers: |

# 9、容器安全，删除特权Pod

|  |
| --- |
| kubectl config use-context KSRS00501  查看此命名空间下的所有 pod，删除有特权 Privileged 或者挂载 volume 的 pod  kubectl get pods XXXX -n production -o yaml | grep -i "privileged: true" #注意冒号后面有一个空格，XXXX换成production命名空间下的pod名。  kubectl get pods XXXX -n production -o jsonpath={.spec.volumes} | jq  或者  **kubectl get pod -n development -o yaml | grep -E "privileged|volumeMounts"**  # 确认启用特权模式的Pod和是否挂载数据卷，记得忽略SA的挂载！  将上面查出来的有特权的pod删除  kubectl delete pod XXXX -n production |

# 10、网络策略 NetworkPolicy

|  |
| --- |
| kubectl config use-context KSRS00501  1 检查namespace标签  # 查看 qaqa 命名空间标签（name: qaqa）  kubectl get ns --show-labels  #打标签  # kubectl label ns qaqa name=qaqa  # kubectl label pod products-service environment=testing -n dev-team  2 创建NetworkPolicy  cat keep-networkpolicy.yaml  apiVersion: networking.k8s.io/v1  kind: NetworkPolicy  metadata:  name: pod-restriction  namespace: test  spec:  **podSelector**:  matchLabels:  app: products-service #根据题目要求的标签修改，这个写的是Pod products-service的标签  policyTypes:  - Ingress #注意，这里只写 - Ingress，不要将 - Egress也复制进来！  ingress:  - from:  - **namespaceSelector**:  matchLabels:  name: qaqa #命名空间有name: qaqa标签的  - from:  - **namespaceSelector: {}** #修改为这样，所有命名空间  podSelector:  matchLabels:  environment: testing #有environment: testing标签的Pod  创建  kubectl apply -f keep-networkpolicy.yaml |

# 11、Trivy 扫描镜像安全漏洞

|  |
| --- |
| kubectl config use-context KSRS00501  1 切换到Master的candidate下  ssh master01  2 获取命名空间kamino 下的所有pod的image  a=`kubectl get pods -n kube-system --output=custom-columns="IMAGE:.spec.containers[\*].image"`  3 检查镜像是否有高危和严重的漏洞  for i in $a;do trivy image --skip-update -s **'HIGH**,**CRITICAL'** $i;done  4 删除有问题的pod  kubectl delete pod XXXX -n kamino |

# 12、AppArmor

|  |
| --- |
| kubectl config use-context KSSH00401  1 切换到node02的root下  ssh node02  sudo -i  2 切换到apparmor的目录，并检查配置文件  cd /etc/apparmor.d/  vi /etc/apparmor.d/nginx\_apparmor  注意，**nginx-profile-3这一行要注释掉**，但要确保profile nginx-profile-3这一行没有注释。  3 执行apparmor策略模块  没有grep到，说明没有启动。  apparmor\_status | grep nginx-profile-3  加载启用这个配置文件  **apparmor\_parser -q /etc/apparmor.d/nginx\_apparmor**    # 再次检查有结果了  apparmor\_status | grep nginx  显示如下内容  nginx-profile-3  4 修改pod文件  vi /cks/KSSH00401/nginx-deploy.yaml  #添加annotations，kubernetes.io/podx名字和containers下的名字一样即可，nginx-profile-3为前面在worker node01上执行的apparmor策略模块的名  字。  annotations:  container.**apparmor**.security.beta.kubernetes.io/**podx**: localhost/**nginx-profile-3**  创建  kubectl apply -f /cks/KSSH00401/nginx-deploy.yaml |

# 13、Sysdig & falco

|  |
| --- |
| kubectl config use-context KSSH00401  1 切换到node02的root下  ssh node02  sudo -i  # 查看容器id  $ **docker ps |grep tomcat**  Sysdig **-M 30 -p** "\*%evt.time,%user.uid,%proc.name" container.id=xxxx>opt/DFA/incidents/summary  sysdig -M 30 -p "%evt.time,%user.uid,%proc.name" --cri /run/containerd/containerd.sock container.name=tomcat123 >> /opt/KSR00101/incidents/summary |

# 14、Pod 安全策略-PSP

|  |
| --- |
| kubectl config use-context KSSH00401  1 切换到Master的root下  ssh master01  sudo -i  2 检查确认apiserver支持PodSecurityPolicy  启用PSP准入控制器（考试中默认已经启用，但以防万一，还是要检查一下的。）  cat /etc/kubernetes/manifests/kube-apiserver.yaml | grep PodSecurityPolicy  确保有下面这行，在1.23的考试中，这一行已经提前给你加上了，但还是需要检查确认一下。  - --enable-admission-plugins=NodeRestriction,**PodSecurityPolicy**  systemctl daemon-reload  systemctl restart kubelet.service  3 创建PSP  cat psp.yaml  apiVersion: policy/v1beta1  kind: **PodSecurityPolicy**  metadata:  name: restrict-policy  namespace: test  spec:  privileged: false  allowPrivilegeEscalation: false  requiredDropCapabilities:  - ALL  hostNetwork: false  hostIPC: false  hostPID: false  runAsUser:  rule: 'RunAsAny'  seLinux:  rule: 'RunAsAny'  supplementalGroups:  rule: 'RunAsAny'  fsGroup:  rule: 'RunAsAny'  volumes:  - '\*'  创建  kubectl apply -f /cks/psp/psp.yaml  4 创建CluserRole和ServiceAccount，并通过ClusterRoleBinding绑定。  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 dany-access-bind --clusterrole=restrict-access-role --serviceaccount=staging:psp-denial-sa |

# 15、启用API server认证

|  |
| --- |
| kubectl config use-context KSCF00301  1 切换到Master的root下  ssh master01  sudo -i  2 确保只有认证并且授权过的REST请求才被允许  - --authorization-mode=**Node,RBAC**  - --enable-admission-plugins=**NodeRestriction** #在1.23考试中，这一个原先为AlwaysAdmit，需要修改为NodeRestriction。  重启kubelet  systemctl daemon-reload  systemctl restart kubelet  3 删除题目要求的角色绑定  查  kubectl get clusterrolebinding system:anonymous  删  **kubectl delete clusterrolebinding system:anonymous** |

# 16、ImagePolicyWebhook容器镜像扫描

|  |
| --- |
| kubectl config use-context KSSH00901  第1步 切换到Master的root下  ssh master01  sudo -i  2步，编辑admission\_configuration.json（题目会给这个目录），修改defaultAllow为false：  vi /etc/kubernetes/epconfig/admission\_configuration.json  **"defaultAllow": false**  #将true改为false  第3步，编辑/etc/kubernetes/epconfig/kubeconfig.yml，添加 webhook server 地址：  certificate-authority: /etc/kubernetes/epconfig/server.crt  **server: https://image-bouncer-webhook.default.svc:1323/image\_policy** #添加webhook server地址  name: bouncer\_webhook  第4步，编辑kube-apiserver.yaml，从官网中引用 ImagePolicyWebhook 的配置信息  - --enable-admission-plugins=NodeRestriction,**ImagePolicyWebhook**  - --admission-control-config-file=/etc/kubernetes/epconfig/admission\_configuration.json #在1.23的考试中，默认这行已经添加了，  # 在kube-apiserver.yaml的 volumeMounts 增加  volumeMounts: #在volumeMounts下面增加  - mountPath: /etc/kubernetes/epconfig  name: **epconfig**  readOnly: true  # 在kube-apiserver.yaml的volumes 增加  volumes: #在volumes下面增加  - name: **epconfig**  hostPath:  path: **/etc/kubernetes/epconfig**  type: **DirectoryOrCreate**  #如果你写的是目录，则是DirectoryOrCreate，如果你写的是文件，则是File  第5步，重启kubelet。  systemctl restart kubelet |