CKS Q&As

Certified Kubernetes Security Specialist (CKS) Exam

Pass Linux Foundation CKS Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.passapply.com/cks.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Linux Foundation Official Exam Center

QUESTION 1

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 retained for 5 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 changes in the namespace kube-system.

3.

Log all other resources in core and extensions at the Request level.

4.

Don\\'t log watch requests by the "system:kube-proxy" on endpoints or

A. See explanation below.

B. PlaceHolder

Correct Answer: A

QUESTION 2

CORRECT TEXT

A PodSecurityPolicy shall prevent the creation 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-restrict-sa in the existing namespace staging.

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

A. See explanation below.

B. PlaceHolder

Correct Answer: A

```
candidate@cli:~$ kubectl config use-context KSMV00102
Switched to context "KSMV00102".
candidate@cli:~$ cat /home/candidate/KSMV00102/pod-security-policy.yaml
---
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: ""
spec:
  seLinux:
    rule: ""
  runAsUser:
    rule: ""
  supplementalGroups: {}
  fsGroup: {}
candidate@cli:~$ vim /home/candidate/KSMV00102/pod-security-policy.yaml
```

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: "prevent-psp-policy"
spec:
  privileged: false
  seLinux:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
```

```
candidate@cli:~$ vim /home/candidate/KSMV00102/pod-security-policy.yaml
candidate@cli:~$ cat /home/candidate/KSMV00102/pod-security-policy.yaml
---
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: "prevent-psp-policy"
spec:
  privileged: false
  seLinux:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00102/pod-security-policy.yaml
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
podsecuritypolicy.policy/prevent-psp-policy created
candidate@cli:~$ cat /home/candidate/KSMV00102/cluster-role.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: ""
rules:
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role.yaml
```

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: "restrict-access-role"
rules:
```

```
candidate@cli:~$ kubectl create clusterrole restrict-access-role --verb=use --resource=psp --dry-run=client -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  creationTimestamp: null
  name: restrict-access-role
rules:
- apiGroups:
  - policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role.yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: "restrict-access-role"
rules:
- apiGroups:
  - policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
```

```
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role.yaml
candidate@cli:~$ kubectl create clusterrole restrict-access-role --verb=use --resource=psp --dry-run=client --resource-name=prevent-psp-policy -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  creationTimestamp: null
  name: restrict-access-role
rules:
- apiGroups:
  - policy
  resourceNames:
  - prevent-psp-policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role.yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: "restrict-access-role"
rules:
- apiGroups:
  - policy
  resourceNames:
  - prevent-psp-policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
```

```
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00102/cluster-role.yaml
clusterrole.rbac.authorization.k8s.io/restrict-access-role created
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ cat /home/candidate/KSMV00102/service-account.yaml
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: "psp-restrict-sa"
  namespace: "staging"
```

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ""
  namespace: ""
candidate@cli:~$ vim /home/candidate/KSMV00102/service-account.yaml
candidate@cli:~$ cat /home/candidate/KSMV00102/service-account.yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: "psp-restrict-sa"
  namespace: "staging"
candidate@cli:~$ kubectl get sa -n staging
NAME      SECRETS   AGE
default   1         6h6m
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00102/service-account.yaml
serviceaccount/psp-restrict-sa created
candidate@cli:~$ kubectl get sa -n staging
NAME              SECRETS   AGE
default           1         6h6m
psp-restrict-sa   1         2s
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ kubectl create clusterrolebinding restrict-access-bind --clusterrole=restri
ct-access-role --serviceaccount=staging:psp-restrict-sa --dry-run -o yaml
W0520 14:41:23.502004   47627 helpers.go:598] --dry-run is deprecated and can be replaced wi
th --dry-run=client.
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  creationTimestamp: null
  name: restrict-access-bind
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: restrict-access-role
subjects:
- kind: ServiceAccount
  name: psp-restrict-sa
  namespace: staging
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role
cluster-role-binding.yaml  cluster-role.yaml
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role
cluster-role-binding.yaml  cluster-role.yaml
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role-binding.yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: restrict-access-bind
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: restrict-access-role
subjects:
- kind: ServiceAccount
  name: psp-restrict-sa
  namespace: staging
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: restrict-access-bind
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: restrict-access-role
subjects:
- kind: ServiceAccount
  name: psp-restrict-sa
  namespace: staging

candidate@cli:~$
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00102/cluster-role-binding.yaml
clusterrolebinding.rbac.authorization.k8s.io/restrict-access-bind created
candidate@cli:~$
```

QUESTION 3

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

A. See the explanation below

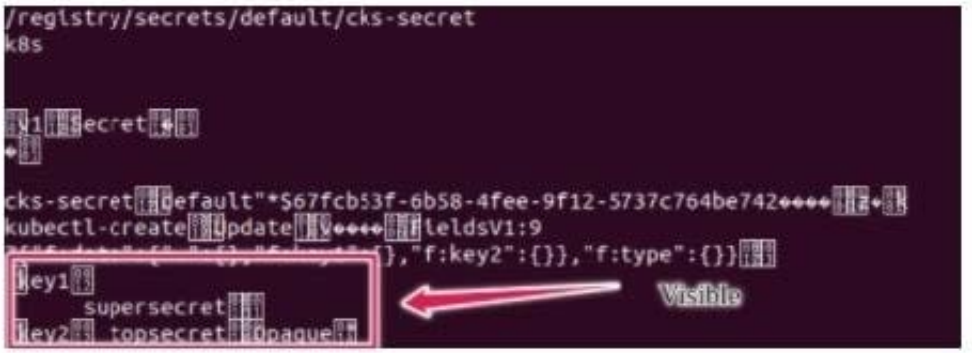
B. PlaceHolder

Correct Answer: A

[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 uk.co.certification.simulator.questionpool.PList@120e0660 [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:backendqarolebinding.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 4

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.

A. See explanation below.

B. PlaceHolder

Correct Answer: A

1.

ETCD secret encryption can be verified with the help of etcdctl command line utility.

2.

ETCD secrets are stored at the path /registry/secrets/$namespace/$secret on the master node.

3.

The below command can be used to verify if the particular ETCD secret is encrypted or not.

ETCDCTL_API=3 etcdctl get /registry/secrets/default/secret1 [...] | hexdump -C

QUESTION 5

```
candidate@cli:~$ kubectl config use-context KSRS00602
Switched to context "KSRS00602".
candidate@cli:~$ ssh ksrs00602-master
Warning: Permanently added '10.240.86.243' (ECDSA) to the list of known hosts.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@ksrs00602-master:~# cat /etc/kubernetes/logpolicy/sample-policy.yaml
---
apiVersion: audit.k8s.io/v1
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"
  # Edit form here below
root@ksrs00602-master:~# vim /etc/kubernetes/logpolicy/sample-policy.yaml
```

```
    - "/api*" # Wildcard matching.
    - "/version"
  # Edit form here below
  - level: RequestResponse
    resources:
    - group: ""
      resources: ["cronjobs"]
  - level: Request
    resources:
    - group: "" # core API group
      resources: ["pods"]
      namespaces: ["webapps"]
# Log configmap and secret changes in all other namespaces at the Metadata level.
  - level: Metadata
    resources:
    - group: "" # core API group
      resources: ["secrets", "configmaps"]

  # A catch-all rule to log all other requests at the Metadata level.
  - level: Metadata
    # Long-running requests like watches that fall under this rule will not
    # generate an audit event in RequestReceived.
    omitStages:
      - "RequestReceived"
```

```
    - "/version"
  # Edit form here below
  - level: RequestResponse
    resources:
    - group: ""
      resources: ["cronjobs"]
  - level: Request
    resources:
    - group: "" # core API group
      resources: ["pods"]
      namespaces: ["webapps"]
# Log configmap and secret changes in all other namespaces at the Metadata level.
  - level: Metadata
    resources:
    - group: "" # core API group
      resources: ["secrets", "configmaps"]

  # A catch-all rule to log all other requests at the Metadata level.
  - level: Metadata
    # Long-running requests like watches that fall under this rule will not
    # generate an audit event in RequestReceived.
    omitStages:
      - "RequestReceived"
root@ksrs00602-master:~# vim /etc/kubernetes/logpolicy/sample-policy.yaml
root@ksrs00602-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
    - command:
        - kube-apiserver
        - --advertise-address=10.240.86.243
        - --allow-privileged=true
        - --audit-policy-file=/etc/kubernetes/logpolicy/sample-policy.yaml
        - --audit-log-path=/var/log/kubernetes/kubernetes-logs.txt
        - --audit-log-maxbackup=1
        - --audit-log-maxage=30
        - --authorization-mode=Node,RBAC
        - --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
```

```
  # A catch-all rule to log all other requests at the Metadata level.
  - level: Metadata
    # Long-running requests like watches that fall under this rule will not
    # generate an audit event in RequestReceived.
    omitStages:
      - "RequestReceived"
root@ksrs00602-master:~# vim /etc/kubernetes/logpolicy/sample-policy.yaml
root@ksrs00602-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
root@ksrs00602-master:~# systemctl daemon-reload
root@ksrs00602-master:~# systemctl restart kubelet.service
root@ksrs00602-master:~# systemctl enable kubelet
root@ksrs00602-master:~# exit
logout
Connection to 10.240.86.243 closed.
candidate@cli:~$
```

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

A. See the explanation below

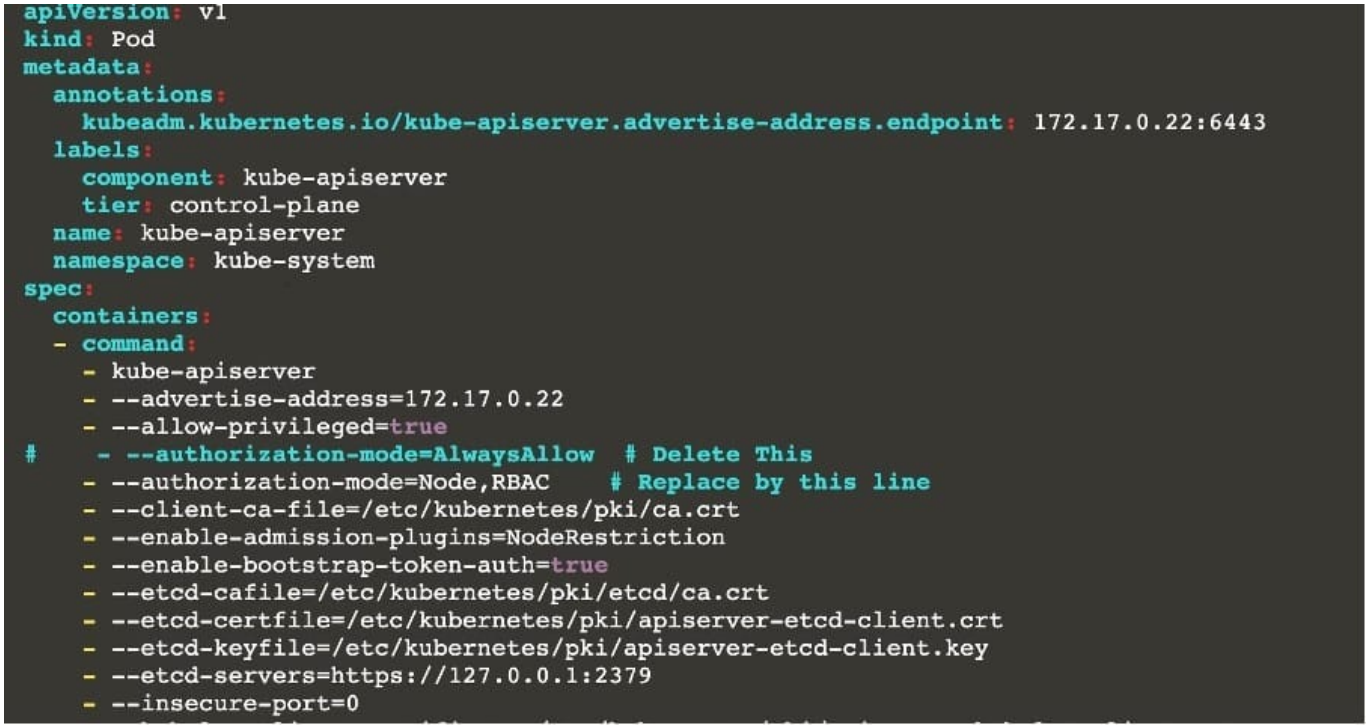
B. PlaceHolder

Correct Answer: A

worker1 $ vim /var/lib/kubelet/config.yaml uk.co.certification.simulator.questionpool.PList@132b77a0 worker1 $ systemctl restart kubelet. # To reload kubelet configssh to master1master1 $ vim /etc/kubernetes/manifests/kube-apiserver.yaml- -- authorizationmode=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



```
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