Problem 12: PodSecurity Policy

- Enable Pod Security policy on the cluster
- Create a PodSecurityPolicy demo to prevent the creation of all privileged pods.
- Create a Role and Role binding to use this policy and assign it to a new service account sam. Do this in a new test namespace.

Solution 12:

Pod Security Policies enable fine-grained authorization of pod creation and updates. PodSecurityPolicy is deprecated in v1.21+ and will be unavailable in v1.25+ but it is still part of CKS certification, so keeping this scenario as it is.

I will be using my killercoda Kubernetes playground(https://killercoda.com/saiyampathak/scenario/kube124) for this example and will be using the same example as mentioned in the kubernetes documentation

First enable the PodSecurityPolicy plugin by changing the `/etc/kubernetes/manifests/kube-apiserver.yaml` file .

```
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=172.17.0.39
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction,PodSecurityPolicy
```

Create a PodSecurityPolicy to deny privileged pods
Note - Before enabling the admission plugin make sure to have a default allow policy so that pods are not prevented

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
```

```
metadata:
  name: demo
spec:
  privileged: false  # Don't allow privileged pods!
  # The rest fills in some required fields.
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  volumes:
  - '*'
```

If you now try to create the privileged pod, it will fail

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod-1
  namespace: default
spec:
  containers:
  - name: centos
    image: centos
    command: ['sh', '-c', 'sleep 999']
    securityContext:
       privileged: true
```

```
Error from server (Forbidden): error when creating "po": pods
"test-pod-1" is forbidden: PodSecurityPolicy: unable to admit
pod: [spec.containers[0].securityContext.privileged: Invalid
value: true: Privileged containers are not allowed]
```

Create service account in test namespace

```
kubectl create ns test
namespace/test created

kubectl create sa sam -n test
serviceaccount/sam created
```

Create a Role and RoleBinding and bind it to SA sam

```
kubectl create role psp --verb=use
--resource=podsecuritypolicy --resource-name=demo -n test
role.rbac.authorization.k8s.io/psp created

kubectl create -n test rolebinding psp-binding --role=psp
--serviceaccount=test:sam
rolebinding.rbac.authorization.k8s.io/psp-binding created
```

No there can be scenarios where you can be asked to create a specific policy and you can control all of the below

Running of privileged containers	`privileged`
Usage of host namespaces	`hostPID` , `hostIPC`
Usage of host networking and ports	`hostNetwork` , `hostPorts`
Usage of volume types	`volumes`
Usage of the host filesystem	`allowedHostPaths`
Allow specific FlexVolume drivers	`allowedFlexVolumes`
Allocating an FSGroup that owns the pod's volumes	`fsGroup`
Requiring the use of a read only root file system	`readOnlyRootFilesystem`
The user and group IDs of the container	`runAsUser` , `runAsGroup` , `supplementalGroups`
Restricting escalation to root privileges	`allowPrivilegeEscalation` , `defaultAllowPrivilegeEscalation`
Linux capabilities	`defaultAddCapabilities` , `requiredDropCapabilities` , `allowedCapabilities`
The SELinux context of the container	`seLinux`
The Allowed Proc Mount types for the container	`allowedProcMountTypes`
The AppArmor profile used by containers	annotations
The seccomp profile used by containers	annotations
The sysctl profile used by containers	`forbiddenSysctls` , `allowedUnsafeSysctls`

Bookmark this page -
https://kubernetes.io/docs/concepts/policy/pod-security-policy/
You can find the examples here.

Example, if you are asked to create a PodSecurityPolicy that allows hostPath volumes only for directory /foo then you can use the below snippet.

```
You can add the following to the PodSecurityPolicy
  allowedHostPaths:
    # This allows "/foo", "/foo/", "/foo/bar" etc., but
    # disallows "/fool", "/etc/foo" etc.
    # "/foo/../" is never valid.
    - pathPrefix: "/foo"
      readOnly: true # only allow read-only mounts
```

This snippet is directly taken from the Kubernetes documentation so make sure you understand the concepts and bookmark the page.