**POD SECURITY POLICIES**

1. **Privileged**
2. **Baseline/default**
3. **Restricted**

**Privileged :**

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

metadata:

name: privileged

annotations:

seccomp.security.alpha.kubernetes.io/allowedProfileNames: '\*'

spec:

privileged: true

allowPrivilegeEscalation: true

allowedCapabilities:

- '\*'

volumes:

- '\*'

hostNetwork: true

hostPorts:

- min: 0

max: 65535

hostIPC: true

hostPID: true

runAsUser:

rule: 'RunAsAny'

seLinux:

rule: 'RunAsAny'

supplementalGroups:

rule: 'RunAsAny'

fsGroup:

rule: 'RunAsAny'

**Baseline or default policy :**

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

metadata:

name: baseline

annotations:

apparmor.security.beta.kubernetes.io/allowedProfileNames: 'runtime/default'

apparmor.security.beta.kubernetes.io/defaultProfileName: 'runtime/default'

seccomp.security.alpha.kubernetes.io/allowedProfileNames: 'docker/default,runtime/default,unconfined'

seccomp.security.alpha.kubernetes.io/defaultProfileName: 'unconfined'

spec:

privileged: false

allowedCapabilities:

- 'CHOWN'

- 'DAC\_OVERRIDE'

- 'FSETID'

- 'FOWNER'

- 'MKNOD'

- 'NET\_RAW'

- 'SETGID'

- 'SETUID'

- 'SETFCAP'

- 'SETPCAP'

- 'NET\_BIND\_SERVICE'

- 'SYS\_CHROOT'

- 'KILL'

- 'AUDIT\_WRITE'

# Allow all volume types except hostpath

volumes:

# 'core' volume types

- 'configMap'

- 'emptyDir'

- 'projected'

- 'secret'

- 'downwardAPI'

- 'persistentVolumeClaim'

- 'awsElasticBlockStore'

- 'azureDisk'

- 'azureFile'

- 'cephFS'

- 'cinder'

- 'csi'

- 'fc'

- 'flexVolume'

- 'flocker'

- 'gcePersistentDisk'

- 'gitRepo'

- 'glusterfs'

- 'iscsi'

- 'nfs'

- 'photonPersistentDisk'

- 'portworxVolume'

- 'quobyte'

- 'rbd'

- 'scaleIO'

- 'storageos'

- 'vsphereVolume'

hostNetwork: false

hostIPC: false

hostPID: false

readOnlyRootFilesystem: false

runAsUser:

rule: 'RunAsAny'

seLinux:

rule: 'RunAsAny'

supplementalGroups:

rule: 'RunAsAny'

fsGroup:

rule: 'RunAsAny'

**Restricted policy :**

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

metadata:

name: restricted

annotations:

seccomp.security.alpha.kubernetes.io/allowedProfileNames: 'docker/default,runtime/default'

apparmor.security.beta.kubernetes.io/allowedProfileNames: 'runtime/default'

seccomp.security.alpha.kubernetes.io/defaultProfileName: 'runtime/default'

apparmor.security.beta.kubernetes.io/defaultProfileName: 'runtime/default'

spec:

privileged: false

# Required to prevent escalations to root.

allowPrivilegeEscalation: false

requiredDropCapabilities:

- ALL

# Allow core volume types.

volumes:

- 'configMap'

- 'emptyDir'

- 'projected'

- 'secret'

- 'downwardAPI'

- 'persistentVolumeClaim'

hostNetwork: false

hostIPC: false

hostPID: false

runAsUser:

# Require the container to run without root privileges.

rule: 'MustRunAsNonRoot'

seLinux:

rule: 'RunAsAny'

supplementalGroups:

rule: 'MustRunAs'

ranges:

# Forbid adding the root group.

- min: 1

max: 65535

fsGroup:

rule: 'MustRunAs'

ranges:

# Forbid adding the root group.

- min: 1

max: 65535

readOnlyRootFilesystem: false

|  |
| --- |
|  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**SELinux** works as a labeling system, which means that all of the files, processes, and ports in a system have an SELinux label associated with them. Labels are a logical way of grouping things together.

**SELinux** defines access controls for the applications, processes, and files on a system. It uses security policies, which are a set of rules that tell SELinux what can or can’t be accessed.

**FSgroup :** The field defines a special supplemental group that assigns a group ID (GID) for all containers in the pod. Also, this group ID is associated with the emptyDir volume mounted at /data/test and with any files created in that volume. You should remember that only certain volume types allow the kubelet to change the ownership of a volume to be owned by the pod. If the volume type allows this (as emptyDir volume type) the owning GID will be the fsGroup.

**runAsUser :** This field specifies the User ID (UID) with which to run the Entrypoint (default executable of the image) of the container process. If the field value is not specified, it defaults to the UID defined in the image metadata. The discussed field can be also used in the spec.containers[].securityContext , in which case it takes precedence over the same field in the PodSecurityContext. In our example, the field specifies that for any containers in the pod, the container process runs with user ID 2500.

**runAsGroup :** This field is useful in cases when you want to run the entrypoint of the container process by a group rather than a user. In this case, you can specify a GID for that group using this field. If the field is not set, the image default will be used. If the field is set both in SecurityContext and PodSecurityContext , the value specified in the container’s SecurityContext takes precedence over the one specified in the PodSecurityContext .

**runAsnonRoot :** The field determines whether the pod’s container should run as a non-root user. If set to true, the kubelet will validate the image at runtime to make sure that it does not run as UID 0 (root) and won’t start the container if it does.

**allowPrivilegeEscalation** : This field controls whether a process can get more privileges than its parent process

**seLinuxOptions** : The [SELinux](https://en.wikipedia.org/wiki/Security-Enhanced_Linux) context to be applied to the container. If the value is unspecified, the container runtime (e.g., Docker) will assign a random SELinux context for each container in a pod.

**capabilities :** Linux capabilities to grant certain root privileges to processes allowing them to run as non-root while giving them root privileges necessary for them to work