**POD SECURITY POLICY**

Pod Security Policies are configurations that define specific security conditions that a pod must meet, in order to be accepted into a cluster. If the conditions are not met, said pod will be rejected.

|  |  |  |  |
| --- | --- | --- | --- |
| SNO | Privileged | Baseline | Restricted |
| 1. | privileged: true | privileged: false | privileged: false |
| 2. | allowPrivilegeEscalation:true | allowPrivilegeEscalation: false | allowPrivilegeEscalation: false |
| 3. | allowedCapabilities:  - '\*' | allowedCapabilities:  defaultset | requiredDropCapabilities:  - ALL |
| 4. | volumes:  - '\*' | volumes: allcorevolumetypes  (except host path volumes) | Volumes : volumes:  - 'configMap'  - 'emptyDir'  - 'projected'  - 'secret'  - 'downwardAPI'  - 'persistentVolumeClaim' |
| 5. | hostNetwork: true | hostNetwork: false | hostNetwork: false |
| 6. | hostPID : true | hostPID : false | hostPID: false |
| 7. | hostIPC:true | hostIPC:false | hostIPC :false |
| 8. | hostPorts:  - min: 0  max: 65535 | Not allowed or minimum restricted to known ports | Not allowed |
| 9. | runAsUser:  rule :RunAsAny | runAsUser:  rule: 'RunAsAny' | runAsUser:  rule: 'MustRunAsNonRoot' |
| 10. | seLinux:  rule: 'RunAsAny' | seLinux:  rule: 'RunAsAny' | seLinux:  rule: 'RunAsAny' |
| 11. | supplementalGroups:  rule: 'RunAsAny' | supplementalGroups:  rule: 'RunAsAny' | supplementalGroups:  rule: 'MustRunAs' |
| 12. | fsGroup:  rule: 'RunAsAny' | fsGroup:  rule: 'RunAsAny' | fsGroup:  rule: ' MustRunAs ' |
| 13. | readOnlyRootFilesystem:true | readOnlyRootFilesystem: false | readOnlyRootFilesystem:false |

Annotations :

For Privileged :

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

For Baseline :

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'

For Restricted :

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'

**Base line PSP :**

|  |  |  |  |
| --- | --- | --- | --- |
| Sno | Rule | Restricted fields in spec | Allowed fields in spec |
| 1. | Host Namespaces | spec.hostNetwork spec.hostPID spec.hostIPC | false |
| 2. | Privileged Containers | spec.containers[\*].securityContext.privileged spec.initContainers[\*].securityContext.privileged | False/undefined |
| 3. | Capabilities | spec.containers[\*].securityContext.capabilities.add spec.initContainers[\*].securityContext.capabilities.add | empty |
| 4. | HostPath Volumes | spec.volumes[\*].hostPath | undefined/nil |
| 5. | Host Ports | spec.containers[\*].ports[\*].hostPort spec.initContainers[\*].ports[\*].hostPort | undefined |
| 6. | SELinux | spec.securityContext.seLinuxOptions spec.containers[\*].securityContext.seLinuxOptions spec.initContainers[\*].securityContext.seLinuxOptions | undefined |

**Step 1 : Create a pod security policy**

**baselinePSP.yaml :**

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'

**Step 2 :** Simply creating a pod security policy does nothing.  In order to use it, the requesting user or target pod's [service account](https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/) must be authorized to use the policy.

Most Kubernetes pods are not created directly by users. Instead, they are created indirectly as part of a [Deployment](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/), [ReplicaSet](https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/), or other templated controller via the controller manager. Instead of assigning the policy to user assign it to pod’s service account. For every pod created by this pod’s service account will use the Pod security policy .

Create a Role or ClusterRole needs to grant access to use the desired policies :

**apiVersion**: rbac.authorization.k8s.io/v1

**kind**: ClusterRole

**metadata**:

**name**: baselinePSProle

**rules**:

- **apiGroups**: ['policy']

**resources**: ['podsecuritypolicies']

**verbs**: ['use']

**resourceNames**:

- baseline

Create a RoleBinding for a role or Cluster Role binding for a cluster Role :

**apiVersion**: rbac.authorization.k8s.io/v1

**kind**: ClusterRoleBinding

**metadata**:

**name**: baselineCRB

**roleRef**:

**kind**: ClusterRole

**name**: baselinePSProle

*apiGroup: rbac.authorization.k8s.io*

**subjects**:

*# Authorize specific service accounts:*

- **kind**: ServiceAccount

**name**: <authorized service account name*>*

*namespace: <authorized pod namespace>*

*# Authorize specific users (not recommended):*

- **kind**: User

**apiGroup**: rbac.authorization.k8s.io

**name**: <authorized user name>

*# Authorize all service accounts in a namespace:*

- **kind**: Group

**apiGroup**: rbac.authorization.k8s.io

**name**: system:serviceaccounts

*# Or equivalently, all authenticated users in a namespace:*

- **kind**: Group

**apiGroup**: rbac.authorization.k8s.io

**name**: system:authenticated

for suppose in the above cluster role binding we have attached our policy to a service account then :

Step – 3 : **podDeployment.yaml**

apiVersion: v1

kind: Pod

metadata:

name: nginx

spec:

securityContext:

runAsUser: 1000

runAsGroup: 3000

fsGroup: 2000

volumes:

- name: demo

emptyDir: {}

containers:

- name: nginx

image: nginx

securityContext:

privileged: false

volumeMounts:

- name: demo

mountPath: /data/demo

capabilities:

add: ["SETUID", "KILL"]

kubectl create deployment nginx --image=nginx

This code will create the pod and assign it to the cluster. And the remaining fields in security context will be default values from the pod security policy.

**podDeployment.yaml**

apiVersion: v1

kind: Pod

metadata:

name: nginx

spec:

securityContext:

runAsUser: 1000

runAsGroup: 3000

fsGroup: 2000

volumes:

- name: demo

emptyDir: {}

containers:

- name: nginx

image: nginx

securityContext:

privileged: false

volumeMounts:

- name: demo

mountPath: /data/demo

capabilities:

add: ["SETUID", "KILL"]

kubectl create deployment nginx --image=nginx

this code will return an error: Error from server (Forbidden): error when creating "STDIN": pods "privileged" is forbidden: unable to validate against any pod security policy: [spec.containers[0].securityContext.privileged: Invalid value: true: Privileged containers are not allowed]

**Restricted PSP :** Everything is taken from the default profile.

|  |  |  |  |
| --- | --- | --- | --- |
| Sno | Rule | Restricted fields in spec | Allowed fields in spec |
| 1. | Volume Types | spec.volumes[\*].hostPath spec.volumes[\*].gcePersistentDisk spec.volumes[\*].awsElasticBlockStore spec.volumes[\*].gitRepo spec.volumes[\*].nfs spec.volumes[\*].iscsi spec.volumes[\*].glusterfs spec.volumes[\*].rbd spec.volumes[\*].flexVolume spec.volumes[\*].cinder spec.volumes[\*].cephFS spec.volumes[\*].flocker spec.volumes[\*].fc spec.volumes[\*].azureFile spec.volumes[\*].vsphereVolume spec.volumes[\*].quobyte spec.volumes[\*].azureDisk spec.volumes[\*].portworxVolume spec.volumes[\*].scaleIO spec.volumes[\*].storageos spec.volumes[\*].csi | undefined/nil |
| 2. | Privilege Escalation | spec.containers[\*].securityContext.allowPrivilegeEscalation spec.initContainers[\*].securityContext.allowPrivilegeEscalation | false |
| 3. | Non-root groups | spec.securityContext.runAsGroup spec.securityContext.supplementalGroups[\*] spec.securityContext.fsGroup spec.containers[\*].securityContext.runAsGroup spec.initContainers[\*].securityContext.runAsGroup | non-zero undefined / nil |