# Kubernetes PSPs

## Understanding and deploying Kubernetes Pod Security Policies

JuanJo Ciarlante @xjjo

Staff Engineer I /  Cross Cloud R&D at VMware

2019-07-02

[ update: video recording at https://youtu.be/LYwD2MVyalw ]

vmware®

# Agenda

Why --The problem

How --Kubernetes approach

How --To implement it

Real world considerations

References

# Why?

If you can **create a Pod**, you **can do anything docker CLI** (or any CRI) could, including **running a privileged container**, using node resources (mount, net, PID), etc

# Ab-using privileged Pods

Scripts from http://bit.ly/jjo-kubectl-root-in-node

```
$ kubectl auth can-i delete node
no - no RBAC policy matched

$ kubectl auth can-i create clusterrolebinding
no - no RBAC policy matched

$ ./kubectl-root-in-host.sh --master
If you don't see a command prompt, try pressing enter.

  ### At the node:
  / # kubectl auth can-i create clusterrolebinding
  yes

  / # kubectl auth can-i delete node
  yes
```

Let's play with privileged Pods, host mounts as a user (SA) *without* cluster-admin

# That greedy Pod ...

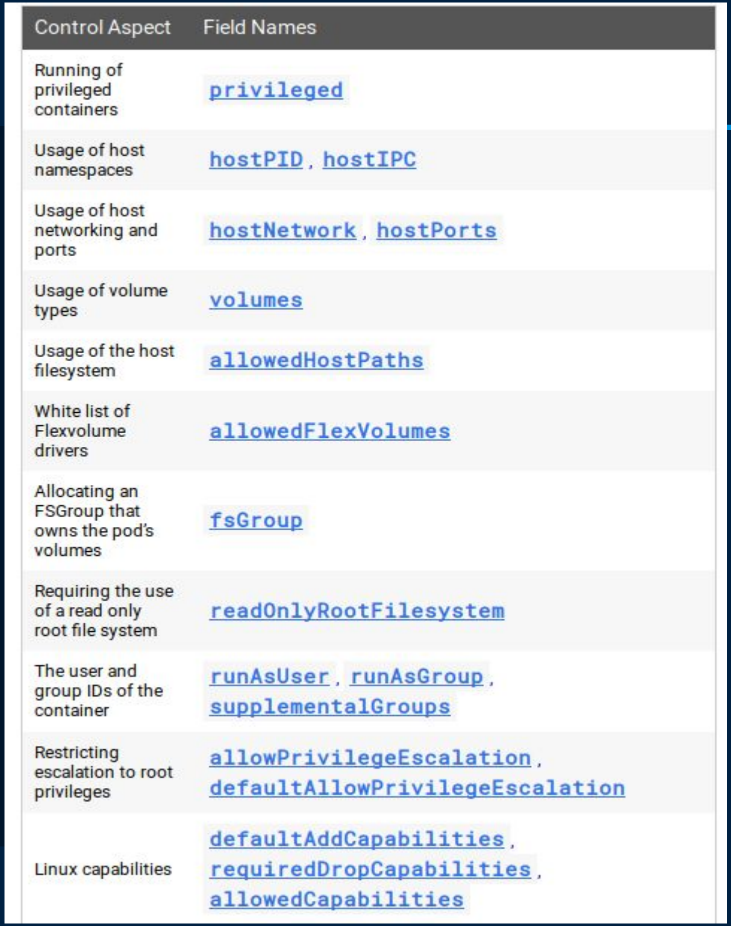
Scripts from https://gist.github.com/jjo/a8243c677f7e79f2f1d610f02365fdd7

```
kubectl run ${podName:?} --restart=Never -it --image overriden --overrides '
{
  "spec": {
    "hostPID": true,
    "hostNetwork": true,
    "tolerations": [{"effect": "NoSchedule","key": "node-role.kubernetes.io/master"}],
    "containers": [
      {
        "name": "alpine",
        "image": "alpine:3.7",
        "command": [
          "nsenter", "--mount=/proc/1/ns/mnt", "--", "/bin/bash"
        ],
        "stdin": true,
        "tty": true,
        "resources": {"requests": {"cpu": "10m"}},
        "securityContext": {
          "privileged": true
        }
      }
    ]
  }
}' --rm --attach "$@"
```

Let's peek at that script

# EHLO Pod Security Policies (PSPs)

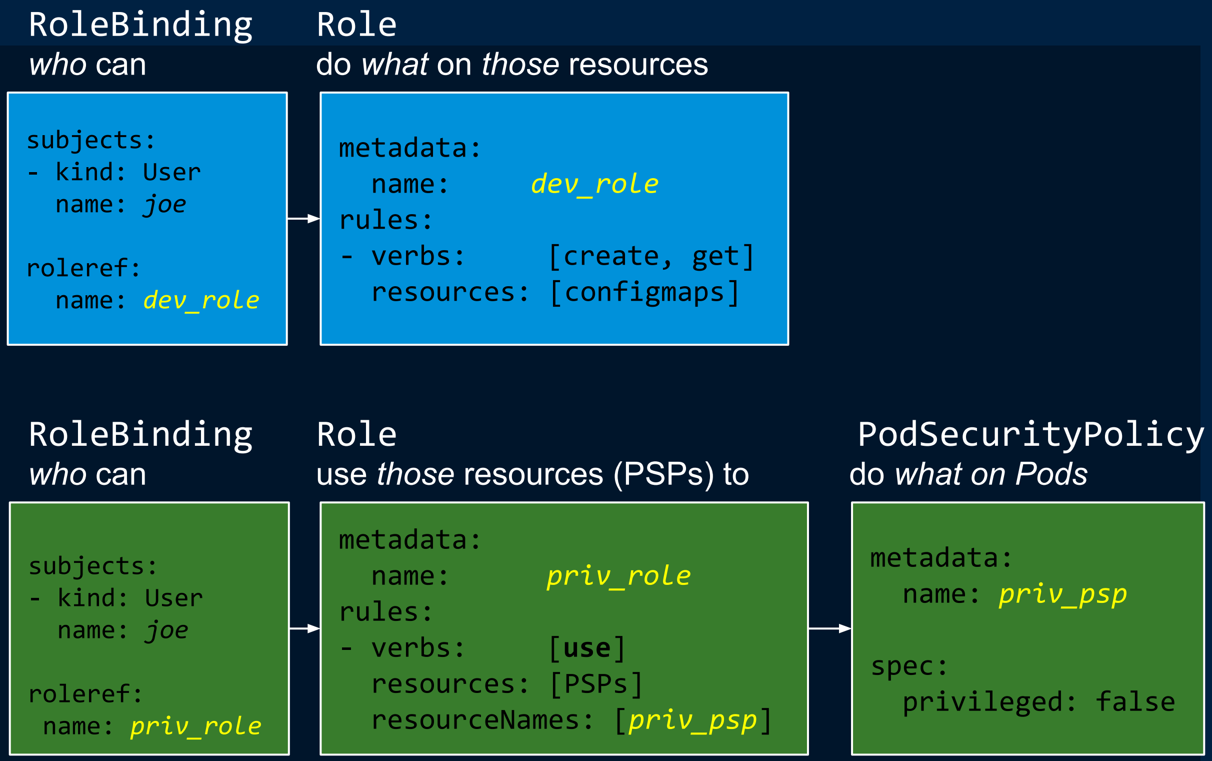
```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
spec:
  allowPrivilegeEscalation: false
  hostIPC: false
  hostNetwork: false
  hostPID: false
  hostPorts: []
  privileged: false
  [...]
  volumes:
  - configMap
  - secret
  - emptyDir
  - projected
  - downwardAPI
  - persistentVolumeClaim
```



| Control Aspect | Field Names |
|---|---|
| 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 |
| White list of 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 |

We want to block those fields, i.e. apply *Policy* to what Pods can specify => **Pod Security Policies**

# How do we *bind* Pods with their PSPs ? => RBAC



`RoleBinding`
*who* can

```
subjects:
- kind: User
  name: joe

roleref:
  name: dev_role
```

`Role`
do *what* on *those* resources

```
metadata:
  name:      dev_role
rules:
- verbs:     [create, get]
  resources: [configmaps]
```

`RoleBinding`
*who* can

```
subjects:
- kind: User
  name: joe

roleref:
 name: priv_role
```

`Role`
use *those* resources (PSPs) to

```
metadata:
  name:       priv_role
rules:
- verbs:      [use]
  resources: [PSPs]
  resourceNames: [priv_psp]
```

`PodSecurityPolicy`
do *what on Pods*

```
metadata:
  name: priv_psp

spec:
  privileged: false
```

## RBAC

Only **one** link leading the ACL:

RoleBinding -> **Role**

## RBAC for PSPs

**Two** links leading to the ACL:

RoleBinding -> Role -> **PSP**

**ERRATUM:**
*priv_psp* should have been called *nopriv_psp*

# … naming × 3 !

- Rolebindings:
  `foo_privileged ?`
  `bar_mayroot ?`
  `baz_nonroot ?`

- Roles:
  `privileged_psp ?`
  `mayroot_psp ?`
  `nonroot_psp ?`

- PodSecurityPolicies [*]:
  `20-privileged ?`
  `40-mayroot ?`
  `60-nonroot ?`

… plus respective filenames 😱

[*] PSPs are also alnum-sort sensitive :),
as there's only **one** final PSP

image: http://devhumor.com/media/when-you-try-to-choose-a-meaningful-variable-name

# Our naming take

## From adding PSPs to all our existing clusters

| PSP | ClusterRole | [Cluster]Rolebinding | namespace | subjects |
|---|---|---|---|---|
| 20-restricted | psp:restricted | (not yet used) | | |
| 40-nonroot | psp:nonroot | (not yet used) | | |
| 60-mayroot | psp:mayroot | [x] psp::mayroot | (any) | system:serviceaccounts |
| 80-privileged | psp:privileged | [ ] psp:kube-system:privileged | kube-system | system:serviceaccounts:kube-system |
| " | " | [ ] psp:kubeprod:privileged | kubeprod | system:serviceaccounts:kubeprod |

# Real world considerations

## From adding PSPs to all our existing clusters

- Note that Pods will be mostly (all?) run by SAs rather than humans
- Using kube-psp-advisor group your "alike" namespaces
- Define a cluster-wide PSP
  - i.e. to be used by all SAs unless having more specific RoleBindings
- Careful with ordering:
  - https://kubernetes.io/docs/concepts/policy/pod-security-policy/#policy-order
- Engage your developers teams on building your PSPs

# Real world considerations

## From adding PSPs to all our existing clusters

- Read how to enable PSPs on cluster platform
- Deploying usually involves:
  - 1- Apply your PSPs *before* enabling them
  - 2- Enable `PodSecurityPolicy` kubeAPI admission controller
  - 3- Recycle your Pods "under control", fix/add PSPs as needed
  - 4- Observe *attached* PSPs via `metadata.annotation["kubernetes.io/psp"]`
    - See `./scripts/report-psps.sh`

# Real world considerations

## From adding PSPs to all our existing clusters

- These slides:
    - bit.ly/jjo-cncf-webinar-psp-19
- Repo used in demos:
    - https://github.com/jjo/jjo-talks/tree/master/2019/cncf-webinar-kube-psps
- Good reads:
    - https://www.cncf.io/wp-content/uploads/2018/07/RBAC-Online-Talk.pdf
    - https://kubernetes.io/docs/concepts/policy/pod-security-policy
    - https://cloud.ibm.com/docs/containers?topic=containers-psp#ibm_psp
    - https://cloud.google.com/kubernetes-engine/docs/how-to/pod-security-policies

# Thank You

vmware®