Research Project 2

Validating replacement filtering features of popular alternative admission controllers for Pod Security Policies

Maarten van der Slik & Frank Wiersma | Wouter Otterspeer (PwC)

kubernetes

Introduction

- Kubernetes
- Pods
- Default: no pod policies
- No limit on configurable pod privileges

- *Demo*

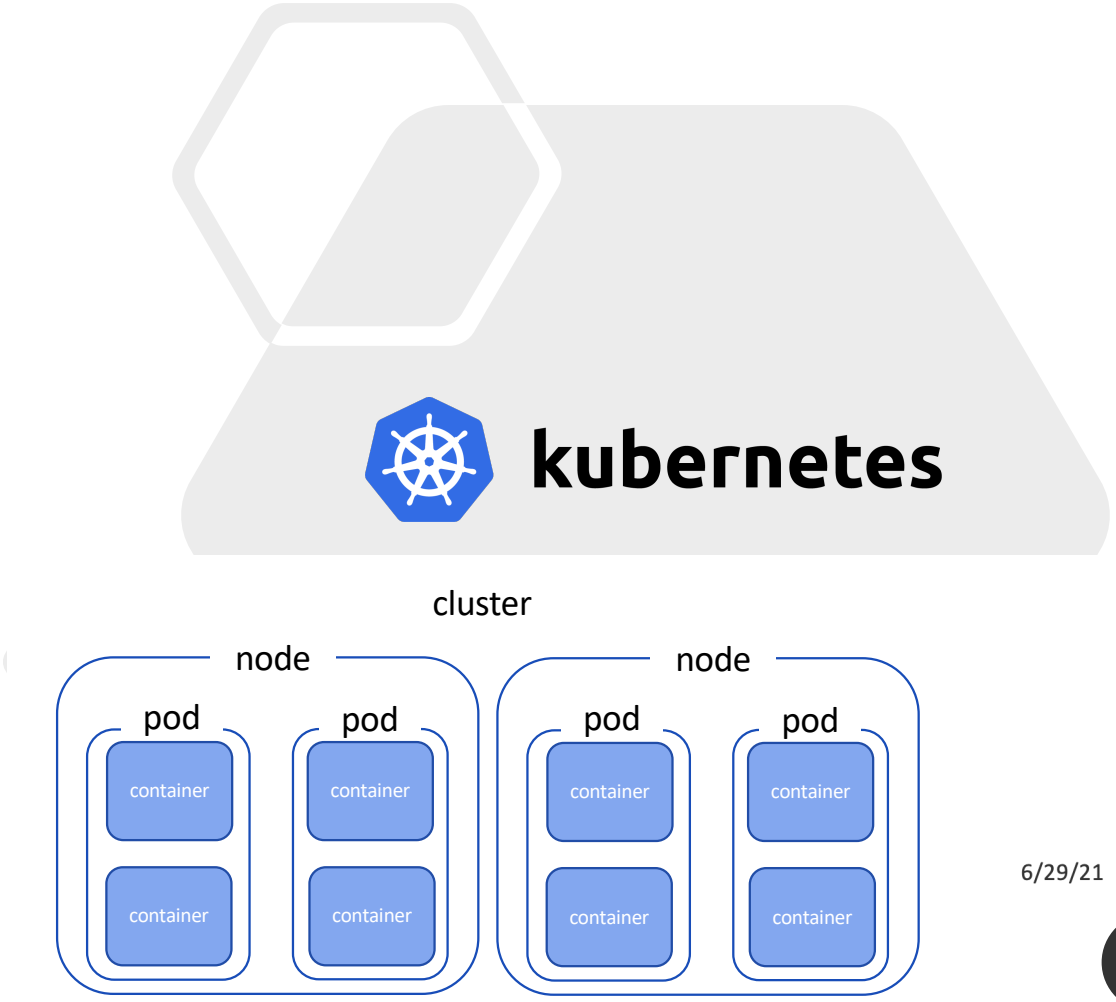
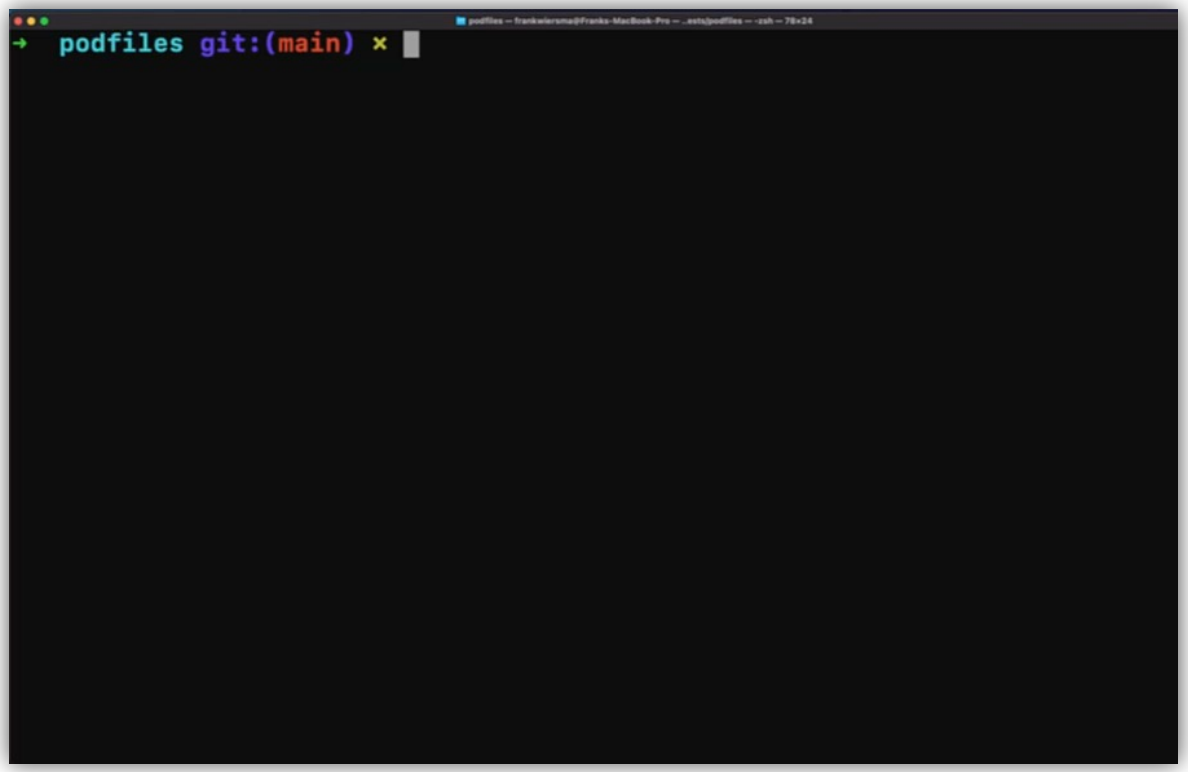


Fig 1. Example of a Kubernetes cluster.

Demo



Introduction

- No limit on configurable pod privileges
  - *Even with RBAC configured*
- PodSecurityPolicies plugin
- Admission Controller for API server

- Deprecated since 1.21
- “Proven confusing” (Sable, 2021)
- Limited replacement feature in development

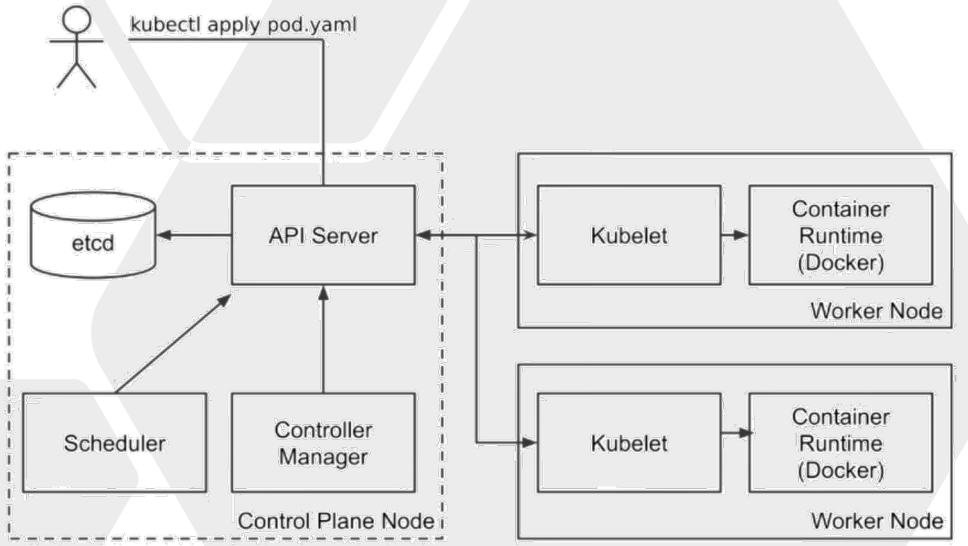


Fig 2. Kubernetes architecture. (Stewart, 2019)

Related Work

- Enhanced isolation of containers by kernel security features (Lin et al., 2018)
- Risks of an improperly configured pod (Artemet al., 2019)
- Kubernetes security best practices (Shamim et al., 2020)
- Purpose & functionality of Pod Security Policies (Bischoff, 2018)
- Defensive methods for Kubernetes clusters (Panagiotis, 2020)

"Are Gatekeeper, Kyverno and k-rail able to cover all filtering functionalities of Pod Security Policies?“

Method

- Assess requirements for alternative admission controllers:
  - Literature research and technical experiments:
    - Map PSP features to Kubernetes API fields
    - Describe PSP operations (validate/mutate)

- Assess if admission controllers can replace PSP:
  - Literature research and technical experiments:
    - Gather built-in features, policy templates and configuration logic
    - Verify conclusions with experiments, when complex logic is required
    - Build and test templates
    - Verify all allowable and disallowable scenarios

Method

Fig 3. Experimental environment

Results - PSP and validating/mutating behavior and fields

PSP Function names	Mutating or Validating	Validates:	Fieldtype
privileged	Validating	spec.containers[].securityContext.privileged	boolean
hostPID	Validating	spec.hostPID	boolean
hostIPC	Validating	spec.hostIPC	boolean
hostNetwork	Validating	spec.hostNetwork	boolean
hostPorts	Validating	spec.containers[].ports.hostPort	integer
volumes	Validating	spec.volumes[]	array
allowedHostPaths	Validating	spec.volumes[].hostPath.path	string
		spec.containers[].VolumeMounts[].readOnly	boolean
allowedFlexVolumes	Validating	spec.volumes[].flexVolume.driver	string
fsGroup MustRunAs	Validating/Mutating**	spec.securityContext.fsgroup	integer
fsGroup MayRunAs	Validating		
fsGroup RunAsAny	Validating		
readOnlyRootFilesystem	Validating	spec.containers[].securityContext.readOnlyRootFilesystem	boolean
runAsUser MustRunAs	Validating/Mutating**	spec.containers[].securityContext.runAsUser	integer
runAsUser RunAsAny	Validating	spec.securityContext.runAsUser	
runAsUser MustRunAsNonRoot	Validating/Mutating***	spec.containers[].securityContext.runAsNonRoot	boolean

Table 1. PSP features

Results - PSP and validating/mutating behavior and fields

PSP Function names	Mutating or Validating	Validates:	Fieldtype
privileged	Validating	spec.containers[].securityContext.privileged	boolean
hostPID	Validating	spec.hostPID	boolean
hostIPC	Validating	spec.hostIPC	boolean
hostNetwork	Validating	spec.hostNetwork	boolean
hostPorts	Validating	spec.containers[].ports.hostPort	integer
volumes	Validating	spec.volumes[]	array
allowedHostPaths	Validating	spec.volumes[].hostPath.path	string
		spec.containers[].VolumeMounts[].readOnly	boolean
allowedFlexVolumes	Validating	spec.volumes[].flexVolume.driver	string
fsGroup MustRunAs	Validating/Mutating**		
fsGroup MayRunAs	Validating		
fsGroup RunAsAny	Validating	spec.securityContext.fsgroup	integer
readOnlyRootFilesystem	Validating	spec.containers[].securityContext.readOnlyRootFilesystem	boolean
runAsUser MustRunAs	Validating/Mutating**	spec.containers[].securityContext.runAsUser	
runAsUser RunAsAny	Validating	spec.securityContext.runAsUser	integer
runAsUser MustRunAsNonRoot	Validating/Mutating***	spec.containers[].securityContext.runAsNonRoot	boolean
runAsGroup MustRunAs	Validating/Mutating**		
runAsGroup MayRunAs	Validating	spec.containers[].securityContext.runAsGroup	
runAsGroup RunAsAny	Validating	spec.securityContext.runAsGroup	integer
supplementalGroups MustRunAs	Validating/Mutating**		
supplementalGroups MayRunAs	Validating		
supplementalGroups RunAsAny	Validating	spec.securityContext.supplementalGroups	integer array
allowPrivilegeEscalation	Validating		boolean
defaultAllowPrivilegeEscalation	Mutating	spec.containers[].securityContext.allowPrivilegeEscalation	boolean
allowedCapabilities	Validating	spec.containers[].securityContext.capabilities.add	string array
requiredDropCapabilities	Mutating	spec.containers[].securityContext.capabilities.add	string array
		spec.containers[].securityContext.capabilities.drop	string array
defaultAddCapabilities	Mutating	spec.containers[].securityContext.capabilities.add	string array
seLinux MustRunAs	Validating/Mutating**	spec.containers[].securityContext.seLinuxOptions	
seLinux RunAsAny	Validating	spec.securityContext.seLinuxOptions	array
allowedProcMountTypes	Validating	spec.containers[].securityContext.procMount	string
annotations: allowedProfileNames	Validating		
annotations: defaultProfileName	Mutating	metadata.annotations	string
annotations: allowedProfileNames	Validating	spec.containers[].securityContext.seccompProfile	
annotations: defaultProfileName	Mutating	spec.securityContext.seccompProfile	array
forbiddenSysctls	Validating		
allowedUnsafeSysctls	Validating	spec.securityContext.sysctls	array

Table 1. PSP features

Results - PSP and Gatekeeper

- Gatekeeper cannot require dropping capabilities and cannot add capabilities
- Gatekeeper misses one template

PSP Function names	Gatekeeper
privileged	T
hostPID	T
hostIPC	T
hostNetwork	T
hostPorts	T*
volumes	T
allowedHostPaths	T
allowedFlexVolumes	T
fsGroup MustRunAs	T,E
fsGroup MayRunAs	T
fsGroup RunAsAny	N/A
readOnlyRootFilesystem	T
runAsUser MustRunAs	T,E
runAsUser RunAsAny	N/A
runAsUser MustRunAsNonRoot	T,E*
runAsGroup MustRunAs	T,E
runAsGroup MayRunAs	T
runAsGroup RunAsAny	N/A
supplementalGroups MustRunAs	E
supplementalGroups MayRunAs	T
supplementalGroups RunAsAny	N/A
allowPrivilegeEscalation	T
defaultAllowPrivilegeEscalation	E
allowedCapabilities	T
requiredDropCapabilities	N
defaultAddCapabilities	N
seLinux MustRunAs	E
seLinux RunAsAny	N/A
allowedProcMountTypes	T
annotations: allowedProfileNames	T
annotations: defaultProfileName	T,E
annotations: allowedProfileNames	T*
annotations: defaultProfileName	T,E
forbiddenSysctls	T
allowedUnsafeSysctls	NT

Table 2. Availability of PSP features in alternative controllers

Results - PSP and Kyverno

- Kyverno cannot implement allowedHostPaths
- Conditional readyonly not available:

```
containers:
  - ...
    volumeMounts:
    - mountPath: /cache
      name: example-volume
      readOnly: true
  volumes:
  - name: example-volume
    hostPath:
      path: /tmp
```

Fig 3. volumeMount example

PSP Function names	Gatekeeper	Kyverno
privileged	T	T
hostPID	T	T
hostIPC	T	T
hostNetwork	T	T
hostPorts	T*	T**
volumes	T	T*
allowedHostPaths	T	N*
allowedFlexVolumes	T	NT
fsGroup MustRunAs	T,E	NT
fsGroup MayRunAs	T	NT
fsGroup RunAsAny	N/A	N/A
readOnlyRootFilesystem	T	T
runAsUser MustRunAs	T,E	NT
runAsUser RunAsAny	N/A	N/A
runAsUser MustRunAsNonRoot	T,E*	NT, TC, N
runAsGroup MustRunAs	T,E	NT
runAsGroup MayRunAs	T	NT
runAsGroup RunAsAny	N/A	N/A
supplementalGroups MustRunAs	E	NT
supplementalGroups MayRunAs	T	NT
supplementalGroups RunAsAny	N/A	N/A
allowPrivilegeEscalation	T	T
defaultAllowPrivilegeEscalation	E	NT
allowedCapabilities	T	NT
requiredDropCapabilities	N	N
defaultAddCapabilities	N	NT
seLinux MustRunAs	E	NT
seLinux RunAsAny	N/A	N/A
allowedProcMountTypes	T	T
annotations: allowedProfileNames	T	T
annotations: defaultProfileName	T,E	N
annotations: allowedProfileNames	T*	T
annotations: defaultProfileName	T,E	NT
forbiddenSysctls	T	NT
allowedUnsafeSysctls	NT	T*

Table 2. Availability of PSP features in alternative controllers

Results - PSP and Kyverno

- Kyverno cannot implement requiredDropCapabilities
- Blacklisting a string is case sensitive

```
securityContext:
  capabilities:
    add:
    - cap_net_raW
    drop:
    - cap_net_raw
    - CAP_NET_RAW
```

Fig 4. Capabilities workaround example

PSP Function names	Gatekeeper	Kyverno
privileged	T	T
hostPID	T	T
hostIPC	T	T
hostNetwork	T	T
hostPorts	T*	T**
volumes	T	T*
allowedHostPaths	T	N*
allowedFlexVolumes	T	NT
fsGroup MustRunAs	T,E	NT
fsGroup MayRunAs	T	NT
fsGroup RunAsAny	N/A	N/A
readOnlyRootFilesystem	T	T
runAsUser MustRunAs	T,E	NT
runAsUser RunAsAny	N/A	N/A
runAsUser MustRunAsNonRoot	T,E*	NT, TC, N
runAsGroup MustRunAs	T,E	NT
runAsGroup MayRunAs	T	NT
runAsGroup RunAsAny	N/A	N/A
supplementalGroups MustRunAs	E	NT
supplementalGroups MayRunAs	T	NT
supplementalGroups RunAsAny	N/A	N/A
allowPrivilegeEscalation	T	T
defaultAllowPrivilegeEscalation	E	NT
allowedCapabilities	T	NT
requiredDropCapabilities	N	N
defaultAddCapabilities	N	NT
seLinux MustRunAs	E	NT
seLinux RunAsAny	N/A	N/A
allowedProcMountTypes	T	T
annotations: allowedProfileNames	T	T
annotations: defaultProfileName	T,E	N
annotations: allowedProfileNames	T*	T
annotations: defaultProfileName	T,E	NT
forbiddenSysctls	T	NT
allowedUnsafeSysctls	NT	T*

Table 2. Availability of PSP features in alternative controllers

Results - PSP and other admission controllers

- Gatekeeper cannot require dropping capabilities and add capabilities
- Gatekeeper misses one template
- Kyverno cannot require dropping capabilities and filter hostpaths correctly
- Kyverno misses 16 templates
- K-rail misses 31 features

PSP Function names	Gatekeeper	Kyverno	K-Rail
privileged	T	T	Y
hostPID	T	T	Y
hostIPC	T	T	N
hostNetwork	T	T	Y
hostPorts	T*	T**	N
volumes	T	T*	N
allowedHostPaths	T	N*	N
allowedFlexVolumes	T	NT	N
fsGroup MustRunAs	T,E	NT	N
fsGroup MayRunAs	T	NT	N
fsGroup RunAsAny	N/A	N/A	N
readOnlyRootFilesystem	T	T	N
runAsUser MustRunAs	T,E	NT	N
runAsUser RunAsAny	N/A	N/A	N
runAsUser MustRunAsNonRoot	T,E*	NT, TC, N	N
runAsGroup MustRunAs	T,E	NT	N
runAsGroup MayRunAs	T	NT	N
runAsGroup RunAsAny	N/A	N/A	N
supplementalGroups MustRunAs	E	NT	N
supplementalGroups MayRunAs	T	NT	N
supplementalGroups RunAsAny	N/A	N/A	N
allowPrivilegeEscalation	T	T	N
defaultAllowPrivilegeEscalation	E	NT	N
allowedCapabilities	T	NT	N
requiredDropCapabilities	N	N	N
defaultAddCapabilities	N	NT	N
seLinux MustRunAs	E	NT	N
seLinux RunAsAny	N/A	N/A	N
allowedProcMountTypes	T	T	N
annotations: allowedProfileNames	T	T	N
annotations: defaultProfileName	T,E	N	N
annotations: allowedProfileNames	T*	T	N
annotations: defaultProfileName	T,E	NT	Y
forbiddenSysctls	T	NT	N
allowedUnsafeSysctls	NT	T*	N

Table 2. Availability of PSP features in alternative controllers

Results - PSP and other admission controllers

- Support other resource types in Gatekeeper:

```
containers[c] {
c = input.review.object.spec.template.spec.containers[_]}
containers[c] {
c = input.review.object.spec.template.spec.initContainers[_]}
```

```
        kinds:
        - apiGroups: ["apps"]
          kinds: ["Deployment"]
```

Fig 5. Field changes in Gatekeepers Rego and yaml templates

Table 2. Availability of PSP features in alternative controllers

PSP Function names	Gatekeeper	Kyverno	K-Rail
privileged	T	T	Y
hostPID	T	T	Y
hostIPC	T	T	N
hostNetwork	T	T	Y
hostPorts	T*	T**	N
volumes	T	T*	N
allowedHostPaths	T	N*	N
allowedFlexVolumes	T	NT	N
fsGroup MustRunAs	T,E	NT	N
fsGroup MayRunAs	T	NT	N
fsGroup RunAsAny	N/A	N/A	N
readOnlyRootFilesystem	T	T	N
runAsUser MustRunAs	T,E	NT	N
runAsUser RunAsAny	N/A	N/A	N
runAsUser MustRunAsNonRoot	T,E*	NT, TC, N	N
runAsGroup MustRunAs	T,E	NT	N
runAsGroup MayRunAs	T	NT	N
runAsGroup RunAsAny	N/A	N/A	N
supplementalGroups MustRunAs	E	NT	N
supplementalGroups MayRunAs	T	NT	N
supplementalGroups RunAsAny	N/A	N/A	N
allowPrivilegeEscalation	T	T	N
defaultAllowPrivilegeEscalation	E	NT	N
allowedCapabilities	T	NT	N
requiredDropCapabilities	N	N	N
defaultAddCapabilities	N	NT	N
seLinux MustRunAs	E	NT	N
seLinux RunAsAny	N/A	N/A	N
allowedProcMountTypes	T	T	N
annotations: allowedProfileNames	T	T	N
annotations: defaultProfileName	T,E	N	N
annotations: allowedProfileNames	T*	T	N
annotations: defaultProfileName	T,E	NT	Y
forbiddenSysctls	T	NT	N
allowedUnsafeSysctls	NT	T*	N

Example template

```yaml
- name: mutate-run-as-non-root-spec-context
  match:
    resources:
      kinds:
      - Pod
  mutate:
    overlay:
      spec:
        containers:
        - (name): "*"
          securityContext:
            (runAsNonRoot): null
        securityContext:
          (runAsNonRoot): null
          +(runAsNonRoot): true
- name: mutate-run-as-non-root-container-conte
  match:
    resources:
      kinds:
      - Pod
  mutate:
    overlay:
      spec:
        containers:
        - (name): "*"
          securityContext:
            +(runAsNonRoot): true
```

```yaml
- name: check-runasgroup
  match:
    resources:
      kinds:
        - Pod
  validate:
    message: >-
      Running with root user IDs is disallowed. The fields
      spec.securityContext.runAsUser, spec.containers[*].securityConte
      and spec.initContainers[*].securityContext.runAsUser must be emp
      or greater than zero.
    pattern:
      spec:
        =(securityContext):
          =(runAsUser): ">0"
        =(initContainers):
          - =(securityContext):
              =(runAsUser): ">0"
        containers:
          - =(securityContext):
              =(runAsUser): ">0"
```

Fig 4. Snippets with three rules of one Kyverno template

Sidenote

- Might publish the repo after RP

https://github.com/
frankwiersma
maartenvdezz

PSP Function names	Gatekeeper	Kyverno	K-Rail
privileged	T	T	Y
hostPID	T	T	Y
hostIPC	T	T	N
hostNetwork	T	T	Y
hostPorts	T*	NT	N
volumes	T	NT	N
allowedHostPaths	T	N	N
allowedFlexVolumes	T	NT	N
fsGroup MustRunAs	T,E	NT	N
fsGroup MayRunAs	T	NT	N
fsGroup RunAsAny	N/A	N/A	N
readOnlyRootFilesystem	T	T	N
runAsUser MustRunAs	T,E	NT	N
runAsUser MustRunAsNonRoot	T,E	NT, N	N
runAsUser RunAsAny	N/A	N/A	N
runAsGroup MustRunAs	T,E	NT	N
runAsGroup MayRunAs	T	NT	N
runAsGroup RunAsAny	N/A	N/A	N
supplementalGroups MustRunAs	E	NT	N
supplementalGroups MayRunAs	T	NT	N
supplementalGroups RunAsAny	N/A	N/A	N
allowPrivilegeEscalation	T	T	N
defaultAllowPrivilegeEscalation	E	NT	N
allowedCapabilities	T	NT	N
requiredDropCapabilities	N	N	N
defaultAddCapabilities	N	NT	N
seLinux MustRunAs	E	NT	N
seLinux RunAsAny	N/A	N/A	N
allowedProcMountTypes	T	T	N
annotations: allowedProfileNames	T	T	N
annotations: defaultProfileName	T,E	N	N
annotations: allowedProfileNames	T	T	N
annotations: defaultProfileName	NT	NT	Y
forbiddenSysctls	T	NT	N
allowedUnsafeSysctls	NT	T	N

Table 3. Templates created

Sidenote

- Might publish the repo after RP

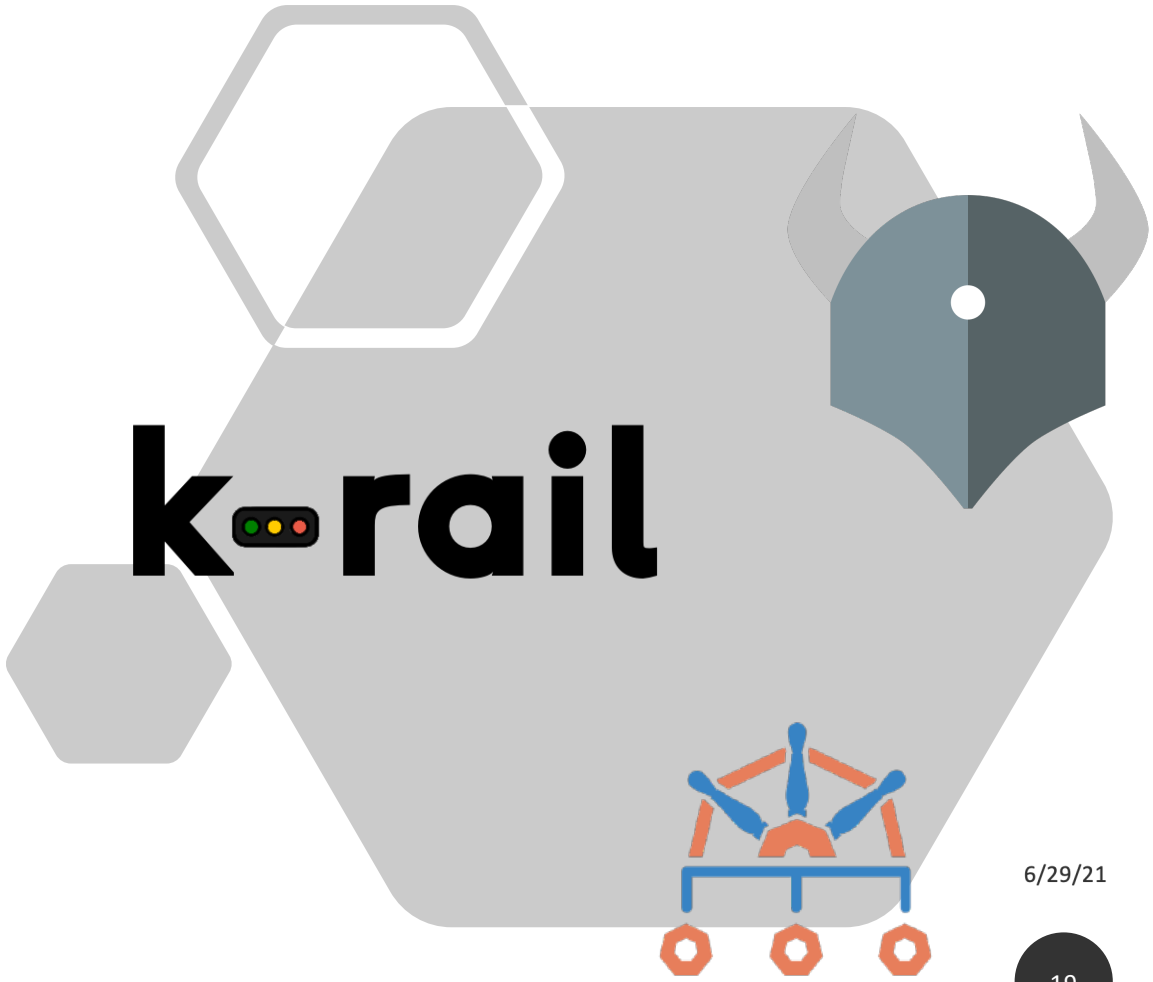
https://github.com/
frankwiersma
maartenvdezz

PSP Function names	Gatekeeper	Kyverno	K-Rail
privileged	T	T	Y
hostPID	T	T	Y
hostIPC	T	T	N
hostNetwork	T	T	Y
hostPorts	T*	TC	N
volumes	T	TC	N
allowedHostPaths	T	N	N
allowedFlexVolumes	T	TC	N
fsGroup MustRunAs	T,E	TC	N
fsGroup MayRunAs	T	TC	N
fsGroup RunAsAny	N/A	N/A	N
readOnlyRootFilesystem	T	T	N
runAsUser MustRunAs	T,E	TC	N
runAsUser MustRunAsNonRoot	T,E	TC	N
runAsUser RunAsAny	N/A	N/A	N
runAsGroup MustRunAs	T,E	TC	N
runAsGroup MayRunAs	T	TC	N
runAsGroup RunAsAny	N/A	N/A	N
supplementalGroups MustRunAs	E	TC	N
supplementalGroups MayRunAs	T	TC	N
supplementalGroups RunAsAny	N/A	N/A	N
allowPrivilegeEscalation	T	T	N
defaultAllowPrivilegeEscalation	E	TC	N
allowedCapabilities	T	TC	N
requiredDropCapabilities	N	N	N
defaultAddCapabilities	N	TC	N
seLinux MustRunAs	E	TC	N
seLinux RunAsAny	N/A	N/A	N
allowedProcMountTypes	T	T	N
annotations: allowedProfileNames	T	T	N
annotations: defaultProfileName	T,E	N	N
annotations: allowedProfileNames	T	T	N
annotations: defaultProfileName	TC	TC	Y
forbiddenSysctls	T	TC	N
allowedUnsafeSysctls	NT	T	N

Table 3. Templates created

Discussion

- Is k-rail able to secure your cluster?
- Is it really necessary to mutate if you can validate?
- Which controller minimizes the attack surface?
- What is the right balance between complexity and flexibility?



Conclusion

- Detailed information is missing in documentation of PSP
- Most of the validation and mutation features can be covered by Gatekeeper and Kyverno
- Kyverno needs a lot of work on templates, but mostly is possible
- Mutations support is limited
- k-rail cannot replace PSP, although it can make your cluster secure

Future Work

- Gatekeeper & Kyverno: research prioritization of policies
- RBAC capabilities in alternative admission controllers
- Mininize security surface outside of the PSP context

References

- Work of Sharma et al.
- Work of Lin et al.
- Work of Bischoff
- Kubernetes PSP documentation & API reference
- Kubernetes SIG security blog post about PSP deprecation
- Kubernetes Enhancement Proposal 2579
- Gatekeeper docs & github
- Kyverno docs & github
- K-rail github

Kubernetes is not secure by default, but using a proper admission controller will surely help.