PSP Replacement: Goals & Requirements

Objective

Replace PodSecurityPolicy without compromising the ability for Kubernetes to limit privilege escalation out of the box. Specifically, create/update pod permission should not be equivalent to root-on-node (or cluster).

Requirements

Requirements with consensus:

- R1. Validating only (i.e. no changing pods to make them comply with policy)
- R2. Safe to enable in new AND upgraded clusters
    a. Dryrun policy changes and/or Audit-only mode
- R3. Built-in in-tree controller
- R4. Capable of supporting Windows in the future, if not in the initial release (xref C3)
    a. Don't automatically break windows pods (xref R2)
- R5. Must be responsive to Pod API evolution across versions
- R6. (fuzzy) Easy to use, don't need to be a kubernetes/security/linux expert to meet the basic objective

Contentious requirements:

- C1. Exceptions or policy bindings by requesting user
- C2. Extensible: should work with custom policy implementations without whole-sale replacement
- C3. Windows support in the initial release (xref R4)
- C4. Support enforcement at the sub-namespace level. E.g. policy applies to some pods in the namespace, and not others.
- C5. Enabled by default
    a. Enforcing anything more conservative than fully privileged by default
- C6. (fuzzy) Powerful and flexible enough for common enterprise use-cases; A viable and simpler alternative to external admission controllers
- C7. Provide an easy migration path from PodSecurityPolicy

Nice to have:

- N1. Enforcement on pod-controller resources (i.e. things embedding a PodTemplate)

Open Questions

- Q1. Where do we draw the policy lines? E.g. are local DoS mitigations in-scope? HostPorts? ReadinessProbes?

Q2. What to do about ephemeral containers?
Q3. How will runtimeclass policy be enforced or not? Should a single namespace be able to have security policy enforced for multiple runtimeclasses (kata and standard pods in the same namespace for instance)?