Kubernetes Security Fundamentals

Pod Security Standards

Pod security standards are crucial for ensuring the security and integrity of applications running in a Kubernetes cluster. These standards define the best practices and guidelines for configuring and managing pods securely, minimizing vulnerabilities and protecting sensitive data.

Pod security standards encompass a range of security measures, including setting security contexts, using PodSecurityPolicies (PSPs), and ensuring compliance with industry standards. By adhering to these standards, you can enhance the security posture of your Kubernetes environment.

Pod Security Standards (PSS) and Pod Security Admission (PSA) are foundational concepts in securing your Kubernetes cluster. They work together to enforce security best practices at the pod level, the fundamental unit of deployment in Kubernetes.

RealLife Example:

Imagine a bustling city with various buildings (pods) housing businesses or residents. Just like any city has building codes and regulations to ensure safety and order, Kubernetes utilizes PSS and PSA to establish security standards for pods within the cluster.

Key Concepts

1. **Security Contexts**
   - Security contexts define security configurations for containers and pods.
   - They control privileges, user IDs, group IDs, and other security settings.
2. **PodSecurityPolicies (PSPs)**
   - PSPs are cluster-level resources that control the security settings of pods.
   - They define a set of conditions that pods must meet to be admitted into the cluster.
3. **Admission Controllers**

◦ Admission controllers are plugins that govern and enforce how the cluster is used.

◦ They can enforce pod security standards by validating and mutating pod specifications.

4. **Compliance Standards**

◦ Ensure pods comply with industry standards and regulatory requirements.

◦ Use tools and policies to enforce compliance across the cluster.

Security Best Practices

1. **Use Security Contexts**

◦ Define security contexts to restrict container privileges and capabilities.

◦ Specify user IDs and group IDs to run containers with non-root users.

2. **Implement PodSecurityPolicies**

◦ Use PSPs to enforce security policies at the cluster level.

◦ Define policies that restrict pod permissions and enforce best practices.

3. **Enable Admission Controllers**

◦ Configure admission controllers to validate and enforce pod security standards.

◦ Use controllers like PodSecurityPolicy, ImagePolicyWebhook, and SecurityContextDeny.

4. **Regular Audits and Compliance Checks**

◦ Regularly audit pods to ensure they comply with security standards.

◦ Use tools like kube-bench and kube-hunter to perform security assessments.

5. **Monitor and Log Pod Activity**

◦ Continuously monitor pod activity to detect anomalies and potential security incidents.

◦ Use logging and monitoring tools to track pod actions and resource usage.

Lab Exercise: Implementing Pod Security Standards

Objective

In this lab, you will learn how to implement pod security standards using security contexts, PodSecurityPolicies, and admission controllers.

Prerequisites

- A running Kubernetes cluster
- kubectl configured to interact with your cluster

Step-by-Step Instructions

Step 1: Define a Security Context

1. **Create a Pod with a Security Context**
   - Define a YAML file for the Pod with a security context.

```
apiVersion: v1
kind: Pod
metadata:
  name: secure-pod
spec:
  containers:
  - name: nginx
    image: nginx
    securityContext:
      runAsUser: 1000
      runAsGroup: 3000
      fsGroup: 2000
      readOnlyRootFilesystem: true
      capabilities:
        drop:
        - ALL
```

2. **Apply the Pod Specification**
   - Create the Pod in the cluster.

```
kubectl apply -f secure-pod.yaml
```

Step 2: Implement PodSecurityPolicies

1. **Create a PodSecurityPolicy**
   - Define a YAML file for the PodSecurityPolicy.

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: restricted-psp
spec:
  privileged: false
  seLinux:
    rule: 'MustRunAs'
  runAsUser:
    rule: 'MustRunAsNonRoot'
  fsGroup:
    rule: 'MustRunAs'
    ranges:
    - min: 1
      max: 65535
  volumes:
  - 'configMap'
  - 'emptyDir'
  - 'persistentVolumeClaim'
```

2. **Apply the PodSecurityPolicy**
   - Apply the policy to the cluster.

```
kubectl apply -f psp.yaml
```

3. **Create a Role and RoleBinding for the PSP**
   - Define a Role that allows using the PSP.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
```

```
  metadata:
    namespace: default
    name: use-psp
  rules:
  - apiGroups: ['policy']
    resources: ['podsecuritypolicies']
    verbs: ['use']
    resourceNames: ['restricted-psp']
```

- Define a RoleBinding to bind the Role to a service account.

```
  apiVersion: rbac.authorization.k8s.io/v1
  kind: RoleBinding
  metadata:
    namespace: default
    name: use-psp
  subjects:
  - kind: ServiceAccount
    name: default
    namespace: default
  roleRef:
    kind: Role
    name: use-psp
    apiGroup: rbac.authorization.k8s.io
```

4. Apply the Role and RoleBinding

- Apply the Role and RoleBinding to the cluster.

```
  kubectl apply -f role.yaml
  kubectl apply -f rolebinding.yaml
```

Step 3: Enable Admission Controllers

1. Configure Admission Controllers

◦ Ensure that the desired admission controllers are enabled in the API Server configuration (usually located in /etc/kubernetes/manifests/kube-apiserver.yaml).

```
    - --enable-admission-plugins=NamespaceLifecycle,LimitRanger,ServiceAccount,DefaultStorageClass,ResourceQuota,PodSecurityPolicy
```

2. **Restart the API Server**
   ◦ Restart the API Server to apply the changes.

Step 4: Regular Audits and Compliance Checks

1. **Use kube-bench for Compliance Checks**
   ◦ Install kube-bench to perform CIS Kubernetes Benchmark checks.

```
    kubectl apply -f https://raw.githubusercontent.com/aquasecurity/kube-bench/main/job.yaml
```

2. **Run kube-bench**
   ◦ Run kube-bench to check compliance with security standards.

```
    kubectl logs -f job/kube-bench
```

Conclusion

Above exercise is just for techies, you can try it out and sort out the errors or perform debugging and troubleshooting yourself it will not come in exam, as it is a Multiple Choice Exam.

By following these steps, you have implemented pod security standards in your Kubernetes cluster. You have defined security contexts, implemented PodSecurityPolicies, enabled admission controllers, and performed regular audits and compliance checks. These practices help ensure the security and integrity of your pods.