Have you hardened your Kubernetes infrastructure?

Hardik Vyas
Product Security @Red Hat

Agenda

- Pod and Container Security
- Network Separation and Hardening
- Authentication and Authorization
- Log Auditing
- Upgrading and Application Security practices

Pod and Container Security

- Configure Security Context for a Pod or Container
- Protecting Pod service account tokens
- Set resource policies for Memory and CPU
- Enable Seccomp and AppArmor or SELinux with appropriate profile
- Appropriate Pod Security Standards policy is applied for all namespaces and enforced
- Images: building, configuring and pulling

A few container `SecurityContext` and `PodSecurityContext` fields to start with:

- runAsUser
- runAsGroup
- runAsNonRoot
- privileged
- allowPrivilegeEscalation
- readOnlyRootFilesystem
- capabilities

The above bullets are not a complete set of security context settings, please refer below for a comprehensive list:

https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.26/#securitycontext-v1-core
https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.26/#podsecuritycontext-v1-core

Specification for illustration purpose only:

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: demo
spec:
 serviceAccountName: demo-sa
 automountServiceAccountToken: false
 securityContext:
   runAsUser: 2000
   runAsGroup: 4000
 containers:
 - name: demo-container
   image: demo-image:latest
   securityContext:
     runAsUser: 3000
     privileged: false
     allowPrivilegeEscalation: false
     capabilities:
       drop:
         - all
       add: [“NET_ADMIN”, “SYS_TIME”]
     readOnlyRootFilesystem: true
   volumeMounts:
     - mountPath: /writeable/location/here
       name: volName
[…]
```

Specifying a memory request and a memory limit:

```yaml
apiVersion: v1
kind: Pod
metadata:
 name: memory-demo
 namespace: memory-example
spec:
 containers:
 - name: memory-demo
   image: demo/image
   resources:
     limits:
       memory: "100Mi"
     requests:
       memory: "50Mi"
```

Images

- Minimize unnecessary content in container images
  - Building a container from scratch or bare minimal base images
- Always pull latest image from a private or trusted registry
- Container images are configured to be run as unprivileged user
  - Configure Dockerfile to include:

```
RUN useradd <user1> && groupadd <group1>
USER <user1>:<group1>
```

- Container images are regularly scanned during creation and in deployment for known vulnerabilities, misconfigurations and outdated libraries
  - CI/CD pipeline

Network Separation and Hardening

- Use network policies and firewalls to separate and isolate resources
  - ingress and egress network policies are applied to all workloads in the cluster
  - default network policies within each namespace, selecting all pods, denying everything, are in place (i.e., create an explicit deny network policy)
  - the Kubernetes API, kubelet API and etcd are not exposed publicly on Internet
- Secure the control plane
  - lock down access to control plane nodes using a firewall and role-based access control (RBAC)
  - use authenticated, encrypted communications using TLS certificates
  - use separate networks for the control plane components and nodes
  - limit access to the Kubernetes etcd server
- Encrypt traffic and sensitive data (such as Secrets) at rest
  - encrypt etcd at rest and use a separate TLS certificate for communication
  - place all credentials and sensitive information encrypted in Kubernetes Secrets rather than in configuration files or ConfigMaps

The following example is a network policy to limit access to the nginx service to Pods with the label access:

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
Metadata:
 name: access-nginx
Spec:
 podSelector:
   matchLabels:
     app: nginx
 Ingress:
 - from:
   - podSelector:
       matchLabels:
         access: "true"
```

A default deny all ingress policy:

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
    name: deny-all-ingress
spec:
    podSelector: {}
    policyType:
    - Ingress
```

A default deny all egress policy:

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
    name: deny-all-egress
spec:
    podSelector: {}
    policyType:
    - Egress
```

Authentication and Authorization

- Use strong user authentication
  - supports X509 Client Certs, OpenID Connect Tokens, Bearer Tokens, etc
- Disable anonymous access (enabled by default)
  - start the kubelet with the `--anonymous-auth=false` flag
- Create RBAC policies with unique roles for users, administrators, developers, service accounts, and infrastructure team
  - only permissions explicitly required for their operation should be used
  - avoid providing wildcard permissions when possible, especially to all resources
  - avoid adding users to the `system:masters` group

Log Auditing

- Enable audit logging (disabled by default)
- Persist logs to ensure availability in the case of node, Pod, or container-level failure
- Configure logging throughout the environment (e.g., cluster API, audit event logs, cluster metric logs, application logs, Pod seccomp logs, repository audit logs, etc.)
- Aggregate logs external to the cluster
- Implement a log monitoring and alerting system tailored to the organization's cluster

For a rule to be considered valid, it must specify one of the four audit levels:

- None
- Metadata
- Request
- RequestResponse

Example:

```yaml
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
  - level: Metadata
[...]
```

Within the Kubernetes environment, some events that administrators should monitor/log include the following:

- API request history
- Performance metrics
- Resource consumption
- Deployments
- Operating system calls
- Protocols, permission changes
- Network traffic
- Pod scaling
- Volume mount actions
- Image and container modification
- Privilege changes / Modifications to RBAC resources
- Scheduled job (cronjob) creations and modifications

Upgrading and Application Security practices

- Keep up with patches, updates, and upgrades
- Perform periodic vulnerability scans and penetration tests
- Uninstall any old unused components
- Switch to alternatives where any software is no longer maintained or deprecated

References:

- NSA & CISA Kubernetes Hardening Guide
- Kubernetes Documentation

Thank you