# Zero Trust Services in Kubernetes

Randy Abernethy,
Managing Partner


**Zero Trust/Least Privilege/Perimeterless/Security in depth**

# Zero Trust

Zero Trust is a model wherein system components trust nothing inside or outside of the perimeter

Cloud Native environments are often (and always should be considered) perimeterless

Everything must be verified before access is granted

# Related concepts

Zero Trust

Perimeterless

Least Privilege

Security In Depth

# Take Away

- Minimize all attack surfaces
- Assume you will be compromised
- Auth all connections/traffic
- Provide software with the smallest set of permissions
- Emit security telemetry
- Understand and control new attack vectors in containerized environments
- Never package unrequired bits
- Never expose unrequired functionality

# Securing Small Services Step by Step

1. Service design and construction
2. Service packaging and container image design
3. Pod specification
4. Platform based Pod policies
5. Network Policy

# 1. Service Design

# Service Design and Construction Considerations

- Metrics and logging that report security events
- Ensure secret data does not leak in metrics/logs
- Log startup (hence restarts), include security context
  - Log the UID/GID the service is using
  - Log capabilities and other syscall perms
  - Log volume mounts
  - Log namespaces
- Log all connections in and out
- Harden metrics, health and readiness endpoints
  - Place on non-app port
- Implement mTLS

# Demo: Service Design

# 2. Container Packaging

# Creating Minimal Container Images

- Build containers
- Scratch containers
- Sanitizing Image Metadata

# Demo: Minimal Containers

# 3. Pod Specification

# Limiting Pod Promiscuity

- Assigning an unprivileged user and group
- Dropping all capabilities
- Avoiding privilege escalation
- Images and sha hashes
- Benefits and dangers of sidecars and init containers

# Demo: Locking down Pods

# 4. Pod Policy

# Pods and Cluster Governance

- Limiting Linux namespaces
- Limiting volume use
- Read only rootfs
- Limiting system calls
- Controlling proc mounts

# Pod Security Policies

| 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` |
| The SELinux context of the container | `seLinux` |
| The Allowed Proc Mount types for the container | `allowedProcMountTypes` |
| The AppArmor profile used by containers | annotations |
| The seccomp profile used by containers | annotations |
| The sysctl profile used by containers | `forbiddenSysctls`, `allowedUnsafeSysctls` |

# Demo: Pod Security Policies

# 5. Network Policy

# Creating Minimal Container Images

- Limiting Ingress
- Limiting Egress
- Using log and trace exceptions to minimize exposure
- Understanding the role of Namespaces

# Demo: Creating a Network Policy

# Thanks!

Related courses:

- Designing microservices for K8s
- Securing Kubernetes
- Designing secure services

randy@rx-m.com
https://rx-m.com/