Kubernetes Security
Zooming In, Zooming Out
A comprehensive Container Security Strategy

Kavya Pearlman
Global Cybersecurity Strategist - Wallarm
@KavyaPearlman | @Wallarm

Rob Richardson
Technical Evangelist - MemSQL
@Rob_Rich | @MemSQL
Introducing Kavya...

Kavya Pearlman

- Well known as the “Cyber Guardian”
- Cybersecurity Strategist at Wallarm
- An Award-winning Cybersecurity Professional
- Founder and CEO of XR Safety Initiative
- Former Information Security Director Linden Lab
- Former Facebook Third Party Security Risk Advisor

Personal interests

Travel, Gaming, Virtual Worlds
Introducing Rob...

Rob Richardson

- Tech Evangelist for MemSQL
- Microsoft MVP
- Leads the Southeast Valley .NET User Group
- AZGiveCamp Organizer

Personal interests

Travel, Coding, and Teaching
Let's Talk About Kubernetes!

- Overview of Containers
- Monolithic vs Microservices
- What is Kubernetes and its Benefits
- Securing K8 - Zooming in
  Essentials to build a secure Kubernetes environment
- Securing K8 - Zooming Out
  Do's and Don'ts for Containerized Environments
- Conclusion
KUBERNETES NEEDS NEW SECURITY MINDSET

Cloud-native applications and infrastructure create several new challenges for all of us security professionals. We need to establish new security programs, have a new mindset and adopt advanced new tools that are focused primarily on securing cloud-native technologies.”

- Kavya Pearlman
Monolith vs. Microservices

User Interface
Business Logic
Data Layer

MONOLITH APPLICATION DB

DATA SOURCE

User Interface
Microservice
Microservice
Microservice

User Interface
Microservice
Microservice
Microservice

DATA SOURCE
DATA SOURCE
DATA SOURCE
Containers vs. VMs

Containers are isolated, but share OS and, where appropriate, bins/libraries.
What is Kubernetes?

Kubernetes Master

API Server

Controller Manager

Scheduler

etcd

Developer/Operator

Users

Kubernetes Node

Kubelet
cAdvisorKube-Proxy

Pod Pod Pod Pod

Pod Pod Pod Pod

Pod Pod Pod Pod

Pod Pod Pod Pod

Kubernetes Node

Kubelet
cAdvisorKube-Proxy

Pod Pod Pod Pod

Pod Pod Pod Pod

Pod Pod Pod Pod

Pod Pod Pod Pod
Benefits of using Kubernetes

- Bring new products to market faster
- Avoid vendor lock-in
- Enjoy peace of mind that your applications are always on
  - Kubernetes self-heals
  - Kubernetes auto-scales
Benefits of using Kubernetes

- It’s the de facto standard for running cloud-native applications at scale
- Free community support or paid professional services
Kubernetes - Zooming In

The Essentials for Building a Secure Kubernetes Environment
Shopify Breach

Caused by lack of K8 security Essentials

Exploited Weakness
API configuration flaw

Type of attack
SSRF Attack whereby metadata used to steal API keys and credential packets

Effect
Thousands of stores and store-clients information was exposed
Caused by lack of K8 security Essentials

Exploited Weakness:
Kubernetes instance and an insecure administrative console

Type of attack
False credentials

Effect
The total scope of the breach is yet unknown
What is Docker?

Dockerfile → Image → Container

docker-compose.yml → Images → Docker swarm
What is Kubernetes?

Docker ecosystem, infographic by Rob Richardson robrich.org
"K8s does not provide a mechanism to enforce security across Namespaces. You should only use it within trusted domains and not use when you need to be able to provide guarantees that a user of the cluster or pods be unable to access any of the other Namespaces resources"

--GCP Team

tl;dr: A namespace is not a security boundary for inter-pod communication.
Roles and ClusterRoles are a whitelist; essentially a list of the allowed permissions.

RoleBindings and ClusterRoleBindings marry users to roles:

- **Subject** includes the person, place, or thing that has been whitelisted.
  
  Ex) a developer, DevOps, a team member, user, or process.

- **Resource** is the kind of object
  
  Ex) pod, service, the cluster itself, or another logic instance related to Kubernetes.

- **Operations** that are whitelisted are action we permit the system to do. It’s an action related to REST method.

- **Namespace** is the kubernetes section that is allowed.
Network Policies

“By default, pods are not isolated; they accept traffic from any source.” - GCP

https://kubernetes.io/docs/concepts/services-networking/network-policies/

- Secure traffic between containers using service mesh tools like Istio
- Disable legacy APIs etcd access from worker nodes (Shopify)
- Restrict API/Dashboard access (Tesla)
Kubernetes: Pod security policies

- Run as non-root user
- Smallest base container
- Don’t install unnecessary software

Note: Don’t run as Root
Configuration Management

Config File in Container
must trust developers, registry, git repo

Environment Variables
Must trust operations

External Key Vault
Must change application

Note: RBAC is usually best
Kubernetes API request lifecycle

1. API HTTP handler
2. Authentication / authorization
3. Mutating admission controllers
4. Object schema validation
5. Validating admission controllers
6. Persisted to ETCD

- Mutating admission webhooks
- Validating admission webhooks
APPLICATION SECURITY

AppSec follows from the above security methods.

Attacks can come from multiple directions. Separate application-specific vulnerabilities

● Orchestrator vulnerabilities
● Container content vulnerabilities
● Client-side elements

You cannot secure Kubernetes without securing applications.

Note: Microservice environments are very useful, but they are not safe without special measures.
Kubernetes - Zooming Out
Do’s and Don’ts for Containerized Environments

**BUILD**
- Artifact Download
- CI/CD pipeline

**DEPLOY**
- Orchestrator
- Container Registries

**RUN**
- Container Runtime Environments
  - Host Runtime
  - Workload at Runtime
DOs for Containerized Environments

- CREATE IMMUTABLE CONTAINERS
- RUN IMAGES ONLY FROM TRUSTED SOURCES
- USE CONTAINER-NATIVE MONITORING TOOLS
NOT To Dos for Containerized Environments

1. Don’t install an operating system in a container
2. Don’t store critical data in a container
3. Don’t put hard-coded credentials for accessing Registry
4. DON’T run a container as root