Zoom In, Zoom Out.

A Fresh look at Kubernetes Security

Rob Richardson
*Technical Evangelist - MemSQL*
@Rob_Rich | @MemSQL

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

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
● How Istio and Service Mesh can affect security
● Conclusion
Kubernetes - Getting started

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

- **User Interface**
- **Microservice**
- **Microservice**
- **Data Source**

Microservice

- **User Interface**
- **Microservice**
- **Microservice**
- **Data Source**
Containers vs. VMs

**VIRTUAL MACHINE**

- **App A**
  - Bins/Libs
  - Guest OS
- **App A’**
  - Bins/Libs
  - Guest OS
- **App B**
  - Bins/Libs
  - Guest OS

**SERVER**

- **Host OS**
- **Hypervisor**

**CONTAINERS**

- **App A**
  - Bins/Libs
- **App A’**
  - Bins/Libs
- **App B**
  - Bins/Libs
- **App B’**
  - Bins/Libs
- **App B’’**
  - Bins/Libs

**Container Orchestrator**

**SERVER**

**Host OS**

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

Kubernetes Master

Developer/Operator

API Server → Controller Manager

Scheduler

etcd

Kubernetes Master

Users

Kubernetes Node

Kubelet → cAdvisor → Kube-Proxy

Pod → Pod → Pod → Pod

Kubernetes Node

Kubernetes Node

Kubelet → cAdvisor → Kube-Proxy

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

The initial point of entry for the Tesla cloud breach, Tuesday’s report said, was an unsecured administrative console for Kubernetes, an open source package used by companies to deploy and manage large numbers of cloud-based applications and resources.
What is Docker?

Dockerfile → Image → Container
docker-compose.yml → Images → Docker swarm

Docker ecosystem, infographic by Rob Richardson robrich.org
What is Kubernetes?

Docker ecosystem, infographic by Rob Richardson robrich.org
Namespaces

“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

- Smallest base container
- 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
Istio Service Mesh
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
What’s next?

Orchestrator vulnerabilities

Container Content Vulnerabilities

Client-side Vulnerabilities
Injection attacks and cross-site scripting

Note: enumerate and secure all the things
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
Open Source Tools For Container Security

- clair
- anchore
- OpenSCAP
- sysdig
- falco
- DAGDA
NOT To Dos for Containerized Environments

- Don't install an operating system in a container
- Don't run unnecessary services
- Don't store critical data in a container
- Don't put hard-coded credentials for accessing Registry
- DON'T run a container as root
Securing Kubernetes with a Service Mesh like Istio

Observe
map, log, discover

Control
access policies, rate limits, a/b testing, canary channel, inject faults, circuit breaker

Secure
mutual tls between containers