Welcome

Intro - demo

Running with scissors - Liz Rice * Least priviledge * Layered security User with permission to create pods, exec into pods Mount host

Make the systems as secure as we can, layer by layer

CNCF 4C - code, container, cluster, cloud

Start with code:

- secure development practices
- keep libraries updated
- sast, dast etc

- TASK: Use trivy to scan menu aggregator - note some errors

Containers

What is a container?

- Namespaced process
  - network, pid, fs (mnt), users etc

- cgroups
- CRI
  - docker, containerd, crio, many others
  - OCI - runc
    - image
    - runtime
    - distribution

  - overlayfs - files remain in layers

compare container and vm

- container - everything running in the same kernel - no separation!
- vm is considered more secure
- containers way faster to start and easier to distribute
- containers cannot simulate other architectures
- vm is expected to change over time, containers not

Container runtimes

- runc - docker, containerd, cri-o
- gvisor
- kata
- firecracker

root

shared kernel - user in container == user outside the container -> root == root part of container runtime - limit default capabilities - privileged containers root == root != root

demo * capabilities in a container * without and with privileged - iptables * as non-root * mounted volume - compare root and non-root

RECOMMENDATION - do not run as root - do not allow to become root - minimise capabilities - be very careful with giving access to host namespaces and volumes

supply chain security

more and more common with attacks against the supply chain do not run unverified containers in production environments (in general be very careful) container signing

container drift

containers are intended to do run only certain commands/do certain things doing unintended things == container drift

software available to prevent container drift - most is commercial non-open source only open source I know of - Neuvector (for Kubernetes)

apparmor, seccomp, selinux

- limit what the containers can do
- runtime default
- optimal security: unique profiles for each software

vulnerability scanning

- verify that the code in our containers is fine
- TASK: use trivy to scan menu-aggregator

->

malicious compliance

- lecture from Kubecon
- trivy and other scanners use heuristics to scan images - e.g. apt, yarn.lock, requirements.txt
- it is possible to make images pass a scan while still containing known security issues
- be careful to not be "malicious by accident", make sure to scan at the right level

building containers

- minimise
  - only what is needed
  - multi-stage builds
  - minimal base containers (alpine, scratch, distroless)
  - remove shells - balance maintenance vs security
  - squash layers
  - slim, dive etc

- no secrets inside the container during build - saved in layers
- automate
- vulnerability scanning
- do not run as root
- after build, containers should be considered immutable

Kubernetes (cluster)

Kubernetes is all about orchestrating containers * everything from the earlier parts apply

Kubernetes is designed for simplicity - allowing as many different containers as possible to run without changes * Starting a container in default k8s is less secure than in default Docker

OWASP top 10

K01: workload conf

Immutable containers

- `spec.template.spec.containers.securityContext.readOnlyRootFilesystem: true`
- Mount emptyDir in folders that need writing (often /tmp)

Run as non-root

```
securityContext:
  runAsUser: 1000
  runAsGroup: 3000
  fsGroup: 2000
  allowPrivilegeEscalation: false
  fsGroupChangePolicy: OnRootMismatch
  privileged: false
  runAsNonRoot: true
```

Apparmor, Seccomp, SELinux

```
securityContext:
  seccompProfile:
    type: RuntimeDefault
  seLinuxOptions:
    level: "s0:c123,c456"
```

```
spec.template.metadata.annotations:
        container.apparmor.security.beta.kubernetes.io/containername: run
```

Enforcement

pre-cluster: kube-score, kubesec, kubeaudit, snyk pod admission: * Pod Security Standards * Kyverno * OPA Gatekeeper

TASK: update a deployment to follow pss/baseline and pss/restricted * start from the initial scissors deployment

Gitops

- automatically applying the wanted state from git repos to the cluster
- Flux, Argo
- Always have the current state in the cluster saved
  - easy recovery, migration

- By requiring PRs/reviews to the synced branch, you also get segregation of duties (two people checking) and non-repudiation (audit trail)
- Fewer people need access to the cluster

K02 Supply chain

- Vulnerability scanning - continous!
- Do not run untrusted containers
    - Container signing
- Secure your CI/CD pipelines

K03 RBAC

- Least priviledge
- Minimise access to what users and services actually need
- cluster-admin == avoid for most cases
- Unique service accounts for each service
    - tokens mounted in pods by default
- GET-LIST-WATCH -> access
- TASK: find out secrets using mounted service account token

K04 Policy as Code

- PaC
- Save your policies in git - permitted by both kyverno and opa gatekeeper
- Kyverno examples

K05 Logging

- Metrics (Prometheus)
- Centralised logging
- Audit logging (API)
- Do not forget to actually analyse the logs!

K06 Authn

- Default user authentication in k8s: certs
    - no way to revoke
- Use other means of authentication, e.g. Tokens in Rancher, OAUTH etc

K07 Network segmentation

- by default all pods can reach all pods
- zero trust
- RECOMMENDATION: default deny, whitelist accepted connections

- TASK: create network policies
    - 3 pods; a can talk to b, b can talk to c;
    - write backends to give quick response

K08 Secrets management

- secrets saved in etcd
    - by default unencrypted

- encryption in etcd == cert in api, encrypts before submitting to etcd
- SealedSecrets - encrypted, key in cluster - can save encrypted secret in git repo
- KMS - Vault - keep secrets in external system, inject into pods
- files are preferred over env variables - env may be part of a dump

K09 Bad K8s conf

- Defaults are becoming more secure
- Worth evaluating the configuration
- CSI-benchmark
    - kube-bench (+ demo)

K10 Outdated K8s

- self-explanatory

Cloud (vms, local systems ...)

- Make sure that the infrastructure is secure
- Hardening
- Remove unused software
- Keep software updated
- CSI-benchmark for e.g. Ubuntu

Final words

Compliance

- do not fulfill compliance requirements because you have to - see it as an opportunity to follow the recommendations by professionals

Layered security

- It is never enough with one layer of security
- Make it hard for the attackers, and force them to constantly meet new obstacles until they give up
- Service -> limited user -> read-only -> isolated network ...