

Security Best Practices

for Amazon Elastic Kubernetes Service (EKS)

Jeremy Cowan, Principal SA Containers



Agenda

How do you solve a problem like containers? EKS Shared Responsibility Model & Security Pillars Security best practices

- Hosts
- Container images
- Identity and Access Management (IAM)
- Network
- Pod and runtime
- Auditing and forensics

Closing thoughts



Challenges posed by containers

Processes running on a shared kernel Isolation implemented by Linux namespaces and cgroups Short lifespans

Traditional/legacy security software is rarely container-aware

- Firewalls
- IDS/IPS
- DLP
- Forensics

Warrants a different approach



Shared Responsibility Model



EKS with self-managed workers

CUSTOMER DATA CONTAINER IMAGES, SOURCE CODE, IAM KUBERNETES POD DISRUPTION BUDGETS POD SECURITY POLICIES RBAC BINDINGS QOS AND PO CLUSTER HPA & VPA PRIORITY NETWORK **CONTROL PLANE EKS CLUSTER CONFIGURATION WORKER NODE SCALING VPC CONFIGURATION** OS, KUBELET, CRI & AMI CONFIGURATION

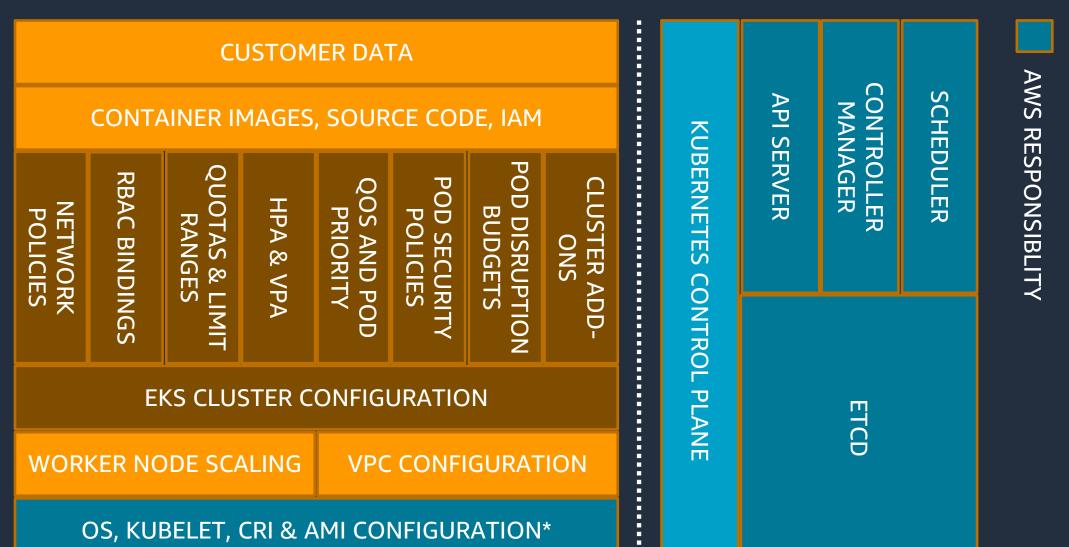
CONTROLLER **SCHEDULER API SERVER** MANAGER



CUSTOMER RESPONSIBILITY

AWS RESPONSIBLITY

EKS with managed node groups (current)





CUSTOMER RESPONSIBILITY

EKS Fargate

CUSTOMER DATA CONTAINER IMAGES, SOURCE CODE, IAM QUOTAS & LIMIT RANGES POD DISRUPTION BUDGETS POD SECURITY
POLICIES RBAC BINDINGS QOS AND POD CLUSTER ADD HPA & VPA NETWORK POLICIES PRIORITY **EKS CLUSTER CONFIGURATION** WORKER NODE SCALING **VPC CONFIGURATION** OS, KUBELET, CRI & AMI CONFIGURATION

CONTROLLER SCHEDULER **API SERVER** MANAGER KUBERNETES **CONTROL PLANE**



AWS RESPONSIBLITY

CUSTOMER RESPONSIBILITY

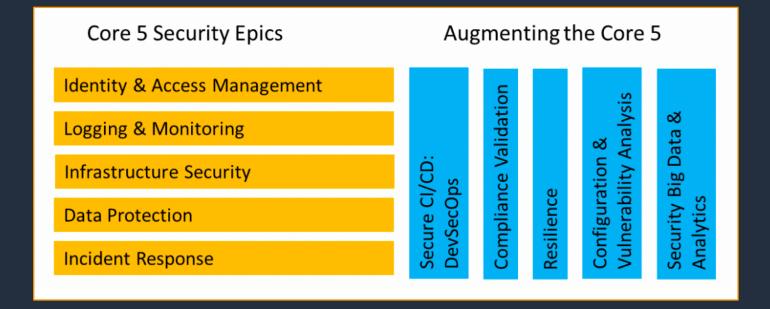
Security Pillars



Container security tenets and epics

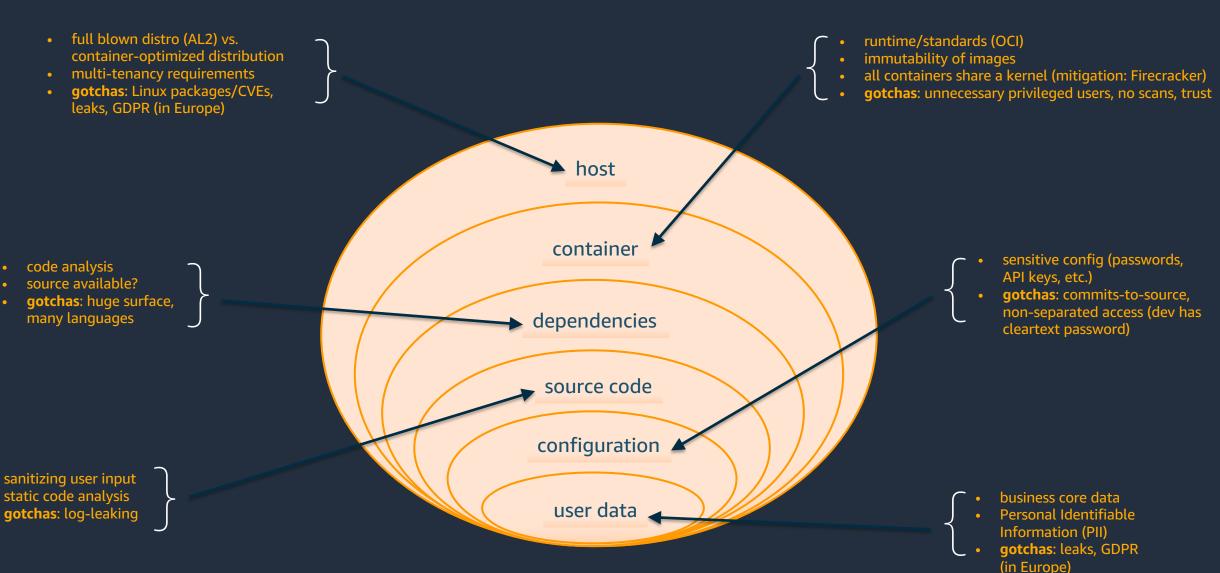
Tenets Epics

- All-encompassing
- Shared responsibility
- Cloud native





Container security onion model: defense in depth





EKS Best Practices for Security



Host Security

Use an OS that is optimized for running containers

- EKS Optimized Amazon Linux 2 & Bottlerocket (preview)
- Alternatives: Atomic, Flatcar Linux, RancherOS

Deploy workers onto private subnets

Run Amazon Inspector to continually assess alignment with best practices and compliance requirements

- kube-bench
- EKS CIS benchmarks

Minimize and audit access

SSM

Use SELinux (RHEL & CentOS)

Audit2Allow, Audit2Why, SEAlert



Securing container images

Scan container images

ECR, Anchore, Clair, Trivy

Use Scratch or a slim base layer

De-fang your images

Remove files with the SETUID and SETGID bits from the image

Always run as a non-root user

Lint your Dockerfiles

Use endpoint policies and private endpoints with Elastic Container Registry (ECR)



Identity and Access Management (IAM)

General guidelines

- Practice the principle of least privilege for AWS IAM and k8s RBAC
- Configure EKS cluster endpoint to be PRIVATE
- Periodically audit access to the cluster

<u>IAM</u>

- Use IRSA to assign AWS identities to pods
- Block access to EC2 metadata

<u>Kubernetes</u>

- Use separate services accounts for each application
- Disable mounting of the default SA token



Network security

Allow all traffic is the default policy

Use k8s network policies for restricting E-W traffic within the cluster

Start with a deny-all global policy and incrementally add policies

- Port 53 (DNS) egress to kube-system
- Allow all within a namespace

Restrict outbound traffic from pods that don't need to connect to external services

SGs for pods & Cilium (L7 policies)

Encrypt service-to-services traffic with a mesh

• Alternatives: Select CNI plug-ins & Nitro instances



Pod and runtime security

Use PSPs or OPA/Gatekeeper to implement runtime security measures:

- Deny privileged escalation
- Deny running as root
- Deny mounting hostPath
- Drop Linux capabilities

Compliment PSPs with AppArmor or Seccomp profiles (if necessary)

Use 3rd party solutions

Aqua, Stackrox, Sysdig Falco, Twistlock



Auditing and forensics

Enable control plane logs

Stream logs from containers to an external log aggregator

Periodically audit Kubernetes control plane and AWS CloudTrail logs for suspicious activity

• Search for the annotations authorization.k8s.io/decision and authorization.k8s.io/reason to ascertain why a call was allow/denied

Immediately isolate pods you suspect have been compromised

- Remove/change labels
- Create network policy to isolate the pod

Cordon the instance (if necessary)

• Capture volatile artifacts on the worker node, e.g. memory, disk, etc.

kube-forensics



Closing thoughts

- Security is everyone's responsibility
- Understand the shared responsibility model
- Shift left & DevSecOps



EKS Best Practices for Security



EKS Best Practices Guide for Security



Q Search

EKS Best Practices Guide for Security

Home

Identity and Access Management

Pod Security

Multi-tenancy

Detective Controls

Network Security

Data Encryption and Secrets

Management

Runtime Security

Infrastructure Security

Regulatory Compliance

Incident Response and Forensics

Image Security

Amazon EKS Best Practices Guide for Security

This guide provides advice about protecting information, systems, and assets that are reliant on EKS while delivering business value through risk assessments and mitigation strategies. The guidance herein is part of a series of best practices guides that AWS is publishing to help customers implement EKS in accordance with best practices. Guides for Performance, Operational Excellence, Cost Optimization, and Reliability will be available in the coming months.

How to use this guide

This guide is meant for security practitioners who are responsible for implementing and monitoring the effectiveness of security controls for EKS clusters and the workloads they support. The guide is organized into different topic areas for easier consumption. Each topic starts with a brief overview, followed by a list of recommendations and best practices for securing your EKS clusters. The topics do not need to read in a particular order.

Understanding the Shared Responsibility Model

https://aws.github.io/aws-eks-best-practices/





Thank you!

