@PEACHYCLOUDSECURITY

AWS

EKS

SECURITY

- EKS GOAT: https://ekssecurity.kubernetesvillage.com/

Peachycloud Security

Seasides Information security conference, Goa

+

=

Amazon EKS

Introduction

Securing Amazon EKS clusters is crucial due to the complexity of cloud environments and the need for proactive security measures.

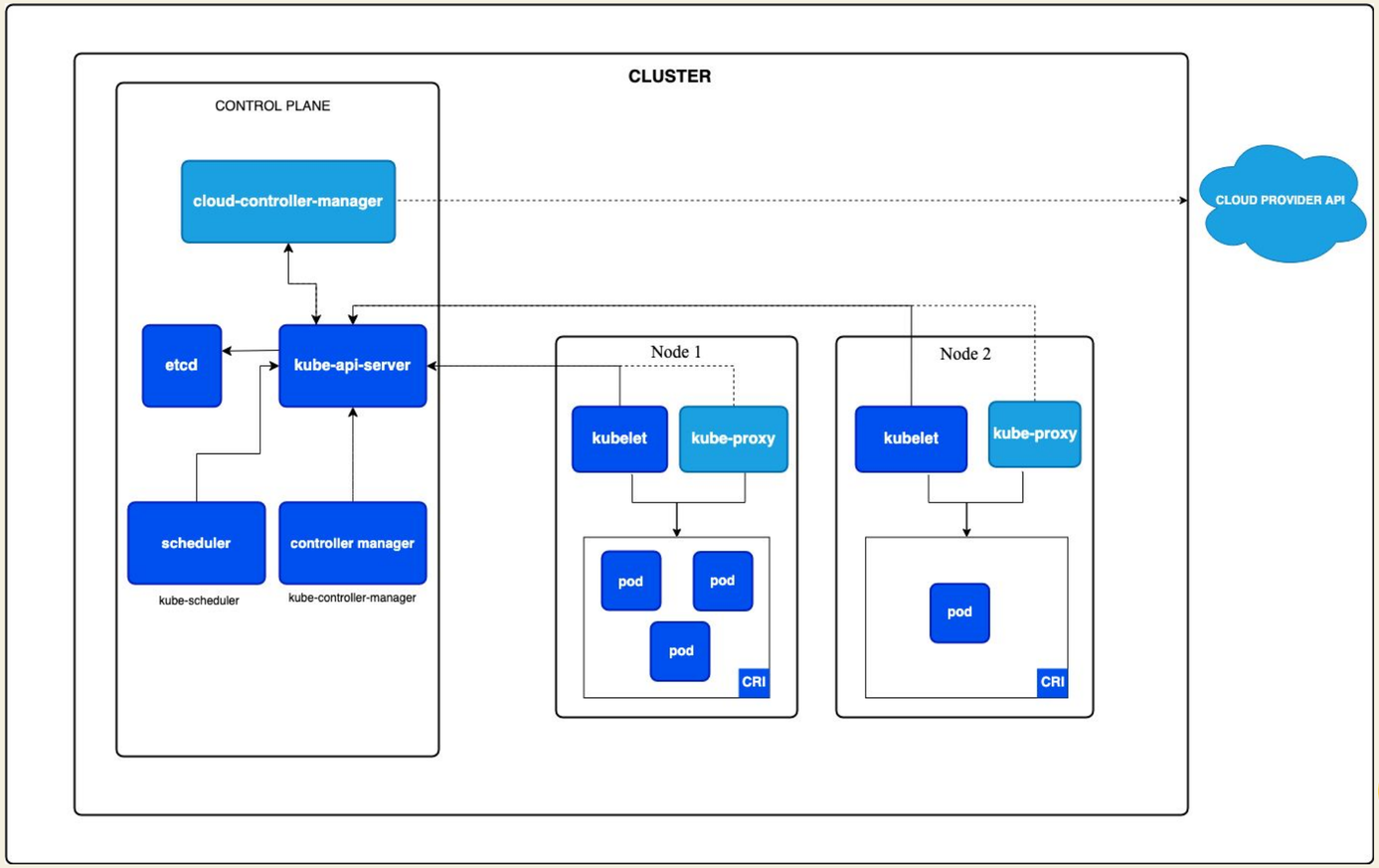
Kubernetes Architecture

Control Plane Component

- kube-apiserver
- etcd
- kube-scheduler
- kube-controller-manager
- cloud-controller-manager

Worker Node Component

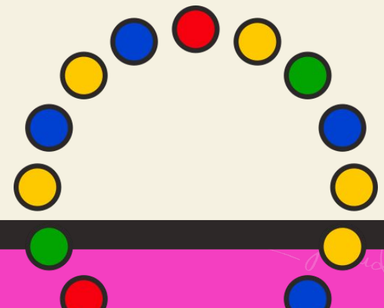
- kubelet
- kube-proxy
- container runtime



EKS Authentication & Authorization

Authentication: Verifies **who** the user is. In AWS EKS, this is managed through **AWS Identity and Access Management (IAM)** or OpenID Connect (OIDC) providers.

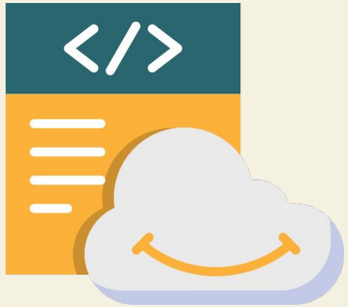
Authorization: Determines **what actions** the authenticated user can perform within the EKS cluster. This is managed through **Kubernetes Role-Based Access Control (RBAC)**



Why Cloud Security Matters?

Cloud providers like AWS are not secure by default; security must be actively managed.

Security OF the Cloud



Cloud Provider (AWS)

CUSTOMER

Security IN the Cloud

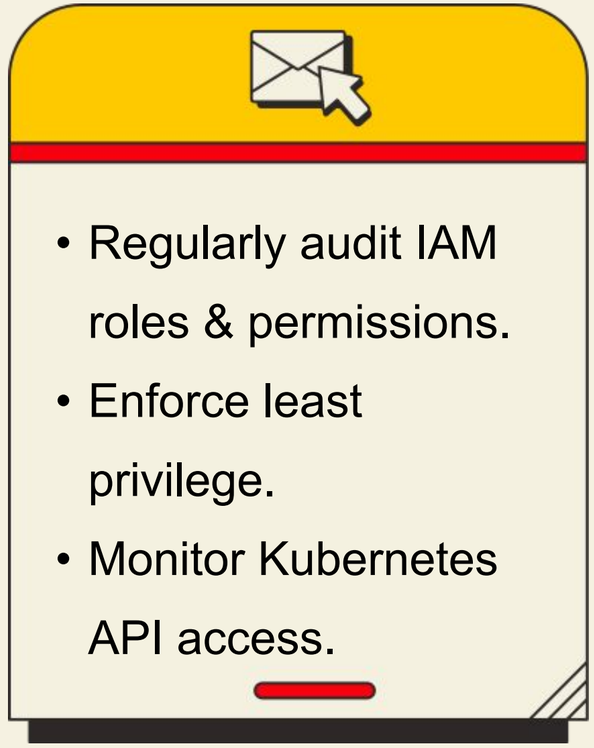
The Shared Responsibility Model requires securing both infrastructure and applications in the cloud.

The Growing Attack Surface

The expanding use of cloud-native applications and Kubernetes (EKS) increases the attack surface, making it essential to understand and mitigate security risks

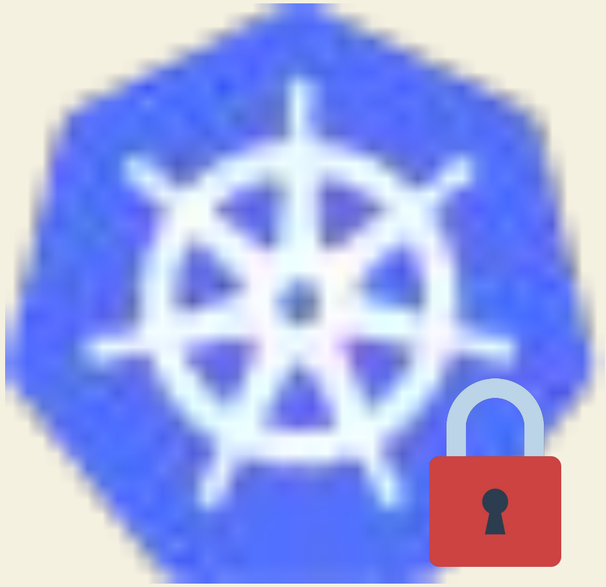
Why EKS Security?

Attacker can exploit a misconfigured EKS cluster, gaining unauthorized access to the Kubernetes API and escalating privileges to compromise sensitive workloads via permissive IAM role.



- Regularly audit IAM roles & permissions.
- Enforce least privilege.
- Monitor Kubernetes API access.

Top 10 EKS Security Practices



By following these top 10 EKS security best practices, you can safeguard your clusters and ensure a robust, secure cloud environment.

Insecure EKS API Server Access

Exposing the Kubernetes API server publicly can lead to unauthorized access.

A publicly accessible API server could allow attackers to interact with your cluster, potentially leading to cluster compromise.

Restrict API server access to specific IP ranges, use private endpoints, and enable role-based access control (RBAC) to limit permissions.

Securing Images from Repositories

Using unverified images from external repositories along with having overly permissive access to ECR can lead to the deployment of compromised images. Attackers could exploit these images to introduce malware or vulnerabilities into the EKS environment.

- Implement strict policies to allow images only from trusted sources.
- Enforce minimal permissions for ECR access
- Automated image scanning before deployment.

Encrypting Data at Rest

If a storage volume is compromised and is unencrypted, it could lead to data breaches where sensitive data is exposed.

Enable encryption on all storage volumes using AWS KMS to ensure that data protection even if the storage is compromised.

Use AWS KMS to encrypt data at rest across services like EBS, EFS, and FSx to protect sensitive information.

Minimise Pod’s IAM Permissions

If a pod's IAM role is overly permissive, an attacker who compromises the pod could exploit these permissions to access sensitive AWS resources, such as S3 buckets or RDS databases, leading to potential data exfiltration or unauthorized modifications.

Audit and limit IAM policies attached to IRSA roles and pod identities, ensuring they follow the principle of least privilege, granting only the permissions necessary for the pod’s specific functions.

Securing Node Group IAM Roles

- Ensure that EKS Cluster node groups are configured with only the permissions necessary for their operation, minimizing the risk.
- Over-permissive IAM roles attached to node groups can allow attackers to exploit these roles to escalate privileges, potentially gaining access to sensitive AWS resources.

Implement the principle of least privilege by carefully auditing and attaching only the necessary policies to node group roles, ensuring they have just enough permissions.

Unsecured Load Balancers

A load balancer configured with overly permissive security settings can expose sensitive services to the internet.

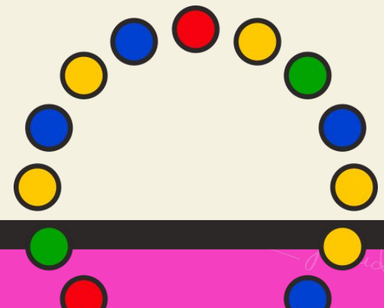
Use security groups and restrict access to load balancers by IP range or VPN.

Exposing services through load balancers without proper security can cause data breaches.

Enforce Network Segmentation

Without network segmentation, an attacker who compromises one pod could move laterally across the network, attacking other pods and escalating privileges.

Implement Kubernetes network policies that enforce least-privilege communication, allowing only necessary traffic between pods.



Realtime Monitoring with GuardDuty

Utilize GuardDuty for real-time threat detection to monitor.

Without proper monitoring, malicious activities such as unauthorized access or privilege escalation can go undetected.

Enable AWS GuardDuty for real-time threat detection to identify suspicious activities in your EKS clusters.

Poor Secrets Management

Storing secrets in plaintext can allow attackers to easily access sensitive information.

Use AWS Secrets Manager to manage secrets.

Inadequate secrets management often involves the exposure of sensitive credentials in configuration files, credentials posing security risk.

Enforce Security with Admission Controller

Use Gatekeeper, an admission controller for Kubernetes, to enforce security policies across your EKS cluster.

Implement admission controller to define and enforce policies that restrict insecure pod configurations, such as preventing privileged containers, enforcing read-only root file systems & disable IMDS v1 hop.

Conclusion

While the top 10 practices are key, also consider securing AWS IAM roles, restricting API endpoints to private access, ensuring node AMIs are up to date, and regularly auditing pod security policies.

EKS GOAT:
https://ekssecurity.kubernetesvillage.com/