Kubernetes Cluster Component Security

API Server
API Server is a critical component of the Kubernetes control plane. It acts as central entity that exposes Kubernets API in Contorl plane.

All Operation on Kubernetes cluster, e.g. deploying application, scaling workload, manage cluster resources.

Authentication : Certificates, Tokens.
Authorization : RBAC = Role Based Access Control (Least Privilege)
Admission Control : PodSecurity, ResourceQuota
Encryption : TLS encryption

Audit Logging: Always keep it Enable

Lab Exercise: Securing the Kubernetes API Server
learn how to configure and secure the Kubernetes API Server. You will enable authentication, set up RBAC, and configure TLS.

Prerequisites
- A running Kubernetes cluster
- kubectl configured to interact with your cluster
- Access to the Kubernetes control plane (API Server configuration)

Step-by-Step Instructions

Step 1: Enable Authentication

1. **Generate Certificates**
   ◦ Use a tool like openssl to generate client certificates.

```
       openssl genrsa -out admin.key 2048
       openssl req -new -key admin.key -out admin.csr -subj
"/CN=admin/O=system:masters"
       openssl x509 -req -in admin.csr -CA ca.crt -CAkey ca.key
-CAcreateserial -out admin.crt -days 365
```

2. **Configure the API Server to Use Certificates**

◦ Edit the API Server manifest (usually located in /etc/kubernetes/manifests/kube-apiserver.yaml).

◦ Add the following flags:

```
- --client-ca-file=/etc/kubernetes/pki/ca.crt
- --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
- --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
```

Step 2: Implement RBAC

1. Create a Role

◦ Define a role with specific permissions.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

2. Create a RoleBinding

◦ Bind the role to a user or group.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: "admin"
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

Step 3: Enable TLS

1. Generate Server Certificates

◦ Similar to client certificates, generate server certificates.

```
      openssl genrsa -out apiserver.key 2048
      openssl req -new -key apiserver.key -out apiserver.csr -subj
"/CN=kube-apiserver"
      openssl x509 -req -in apiserver.csr -CA ca.crt -CAkey ca.key
-CAcreateserial -out apiserver.crt -days 365
```

2. Configure the API Server to Use TLS

◦ Ensure the --tls-cert-file and --tls-private-key-file flags are set with the correct paths in the API Server manifest.

Step 4: Audit Logging

1. Enable Audit Logs

◦ Configure the API Server to log audit events.

```
      - --audit-log-path=/var/log/kubernetes/audit.log
      - --audit-policy-file=/etc/kubernetes/audit-policy.yaml
```

2. Define an Audit Policy

◦ Create a policy file to specify what events to log.

```
      apiVersion: audit.k8s.io/v1
      kind: Policy
      rules:
      - level: Metadata
        resources:
        - group: ""
          resources: ["pods"]
```

3. Apply the Audit Policy

◦ Place the audit policy file in the specified path and restart the API Server.

Above exercise is just for techies, you can try it out and sort out the errors or perform debugging yourself it will not come in exam, as it is Multiple Choise Exam.
By following these steps, you have configured and secured the Kubernetes API Server. You have enabled authentication, implemented RBAC, configured TLS for secure communication, and set up audit logging. These practices help protect the API Server from unauthorized access and provide insights into cluster activities for security and compliance.