Kubernetes Threat Model

Access to Sensitive Data

Access to sensitive data is a critical aspect of Kubernetes security. Sensitive data includes secrets, configuration files, credentials, and any other information that, if exposed, could compromise the security and integrity of applications and the cluster.

Kubernetes provides mechanisms to manage and protect sensitive data, such as Secrets and ConfigMaps. Properly configuring these resources and implementing strict access controls can help prevent unauthorized access and ensure data security.

RealLife Example:

Imagine a grand treasure vault within the kingdom, storing priceless riches and confidential documents. Just as the king entrusts only the most loyal subjects to access these valuables, securing access to sensitive data in Kubernetes requires careful consideration.

Key Concepts

1. **Secrets**
    - Kubernetes Secrets store sensitive data, such as passwords, API keys, and certificates.
    - Secrets can be used as environment variables or mounted as volumes in pods.
2. **ConfigMaps**
    - ConfigMaps store configuration data that is not sensitive.
    - They can be used to decouple configuration artifacts from container images.
3. **Role-Based Access Control (RBAC)**
    - RBAC controls who can access and manipulate Kubernetes resources.
    - Use RBAC to restrict access to sensitive data based on user roles.
4. **Encryption**
    - Encrypt sensitive data at rest and in transit to protect against unauthorized access.

    ◦ Kubernetes supports encryption of Secrets at rest.

5. **Audit Logging**
    ◦ Enable audit logging to track access and modifications to sensitive data.
    ◦ Regularly review logs to detect unauthorized access attempts.

Security Best Practices

1. **Use Kubernetes Secrets for Sensitive Data**
    ◦ Store sensitive data in Secrets rather than ConfigMaps.
    ◦ Mount Secrets as volumes or use them as environment variables in pods.
2. **Implement RBAC for Access Control**
    ◦ Define roles and role bindings to restrict access to Secrets and ConfigMaps.
    ◦ Use the principle of least privilege to grant only necessary permissions.
3. **Encrypt Sensitive Data**
    ◦ Enable encryption at rest for Secrets.
    ◦ Use TLS to encrypt data in transit between Kubernetes components.
4. **Enable and Monitor Audit Logging**
    ◦ Configure audit logging to track access to sensitive data.
    ◦ Set up alerts for suspicious activities and potential breaches.
5. **Regularly Rotate Secrets and Credentials**
    ◦ Implement a process for regularly rotating Secrets and credentials.
    ◦ Update applications to use new Secrets without downtime.