Kubernetes Security Fundamentals

Authentication

Authentication in Kubernetes is crucial for verifying the identity of users and components interacting with the Kubernetes API. It ensures that only authorized entities can access and manage the cluster resources, providing a foundational layer of security.

Kubernetes supports multiple authentication mechanisms to verify the identity of users and components. These mechanisms include client certificates, bearer tokens, OpenID Connect (OIDC), and more. Properly configuring authentication helps secure the Kubernetes API and ensures that only trusted entities can interact with the cluster.

Real Life Example:

Imagine a well-fortified city with a robust gatehouse. Just like the guards at the gatehouse meticulously check the credentials of visitors before allowing entry, authentication in Kubernetes ensures only authorized users can interact with the cluster control plane.

Key Concepts

1. **Client Certificates**
   - Certificates issued to users and components to authenticate their identity.
   - Managed by a Certificate Authority (CA).
2. **Bearer Tokens**
   - Tokens used for authentication, typically short-lived and associated with a user or service account.
   - Can be static tokens or dynamically issued tokens (e.g., via OpenID Connect).
3. **OpenID Connect (OIDC)**
   - A standard protocol for authenticating users using an external identity provider.
   - Integrates with OAuth 2.0 to provide tokens for API access.
4. **Service Accounts**

◦ Special accounts used by applications and components running within the cluster.

◦ Associated with bearer tokens to authenticate API requests.

Security Best Practices

1. **Use Strong Authentication Methods**
   ◦ Prefer client certificates or OIDC for user authentication.
   ◦ Avoid using static bearer tokens for long-term access.
2. **Rotate Credentials Regularly**
   ◦ Implement policies for rotating client certificates and tokens regularly.
   ◦ Ensure that expired or compromised credentials are promptly revoked.
3. **Limit Privileges with RBAC**
   ◦ Use Role-Based Access Control (RBAC) to limit the permissions of authenticated entities.
   ◦ Apply the principle of least privilege to grant only necessary access.
4. **Enable Audit Logging**
   ◦ Enable audit logging on the API Server to track authentication events.
   ◦ Regularly review logs to detect unauthorized access attempts.
5. **Secure Communication**
   ◦ Use TLS to encrypt communication between clients and the API Server.
   ◦ Ensure that all certificates and keys are securely managed and stored.

Note: Follow the official Documentation for lab practice, no exclusive Lab Exercise for Authentication and Authorization.