Mastering Kubernetes Security: From Containers to Cluster Fortresses

Deepu K Sasidharan
Staff Developer Advocate @ Okta

Understanding Kubernetes Security

Transport security

All API communication is done via TLS using valid certificates

Authentication

All API requests are authenticated with one of the several authentication mechanisms supported by Kubernetes

Authorization

All authenticated requests are authorized using one or more of the supported authorization models

Admission control

All authorized requests, except read/get requests, are validated by admission control modules

Kubernetes Security Best practices

https://a0.to/k8s-security-best-practices

Use RBAC

Why?

- Most secure Authorization mechanism for Kubernetes
- Most widely used and most flexible
- Ideal for enterprise and medium-large orgs
- Easy to model business rules

How?

- Check if RBAC is enabled
  - `kubectl cluster-info dump | grep authorization-mode`
- Use `--authorization-mode` flag for the API server to enable RBAC
- Create Role/ClusterRole and RoleBinding/ClusterRoleBinding as required

How?

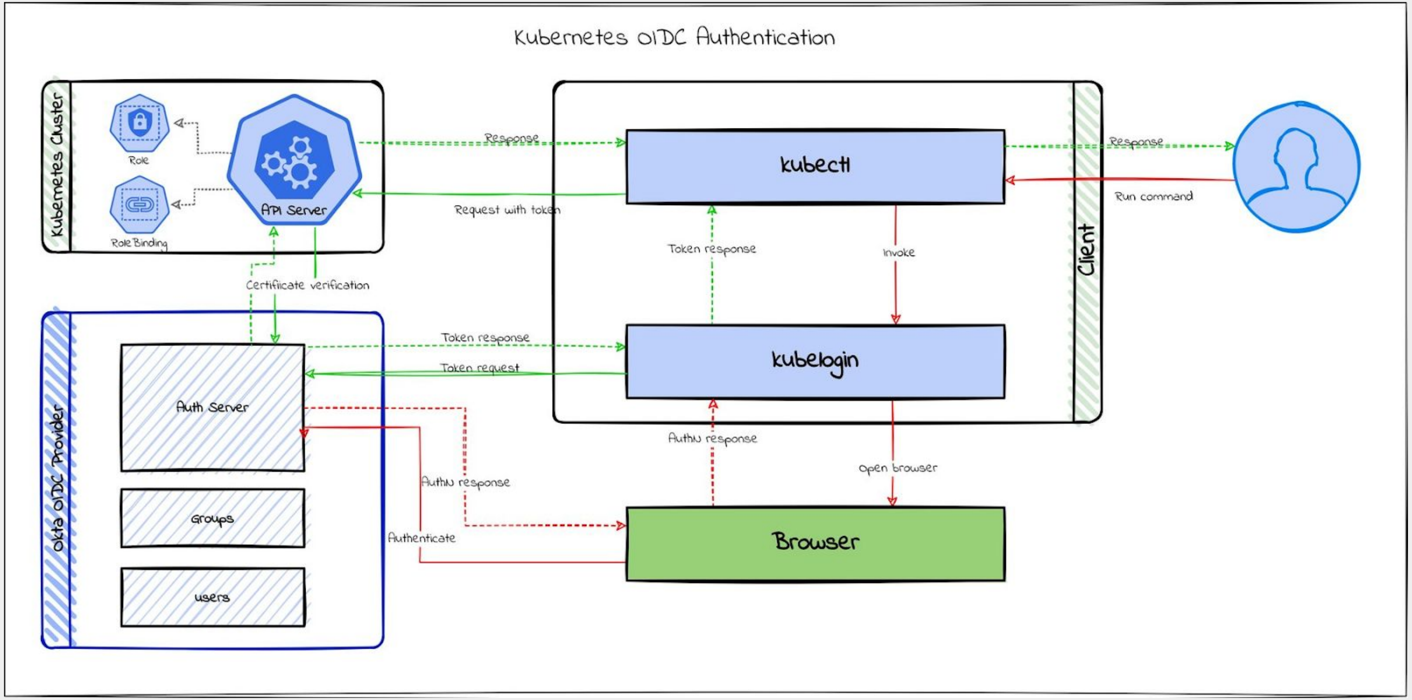
```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: fancy-namespace
  name: pod-service-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods", "services”]
  verbs: ["get", "watch", "list"]

—

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods-services
  namespace: fancy-namespace
roleRef:
  kind: Role #this must be Role or ClusterRole
  name: pod-service-reader  # this must match the name of the Role or ClusterRole you wish to bind to
  apiGroup: rbac.authorization.k8s.io
subjects: # subject can be individual users or a group of users. Group is defined in the external
authentication service, in this case, an OIDC server
  - kind: Group
    name: k8s-restricted-users
```

Use OpenID Connect

Why?

- Most secure Authentication mechanism
- Most scalable
- Ideal for clusters accessed by large teams as it provides a single sign-on solution
- Easy to onboard and offboard users

How?



Kubernetes OIDC Authentication
Kubernetes Cluster
Role
Role Binding
API Server
Response
Request with token
Certificate verification
kubectl
Response
Run command
Token response
Invoke
Client
Okta OIDC Provider
Auth Server
Groups
users
Token response
Token request
kubelogin
AuthN response
Authenticate
AuthN response
Open browser
Browser

How?

How to Secure Your Kubernetes Cluster with OpenID Connect and RBAC

https://a0.to/k8s-api-server-oidc

Use Secure Secrets

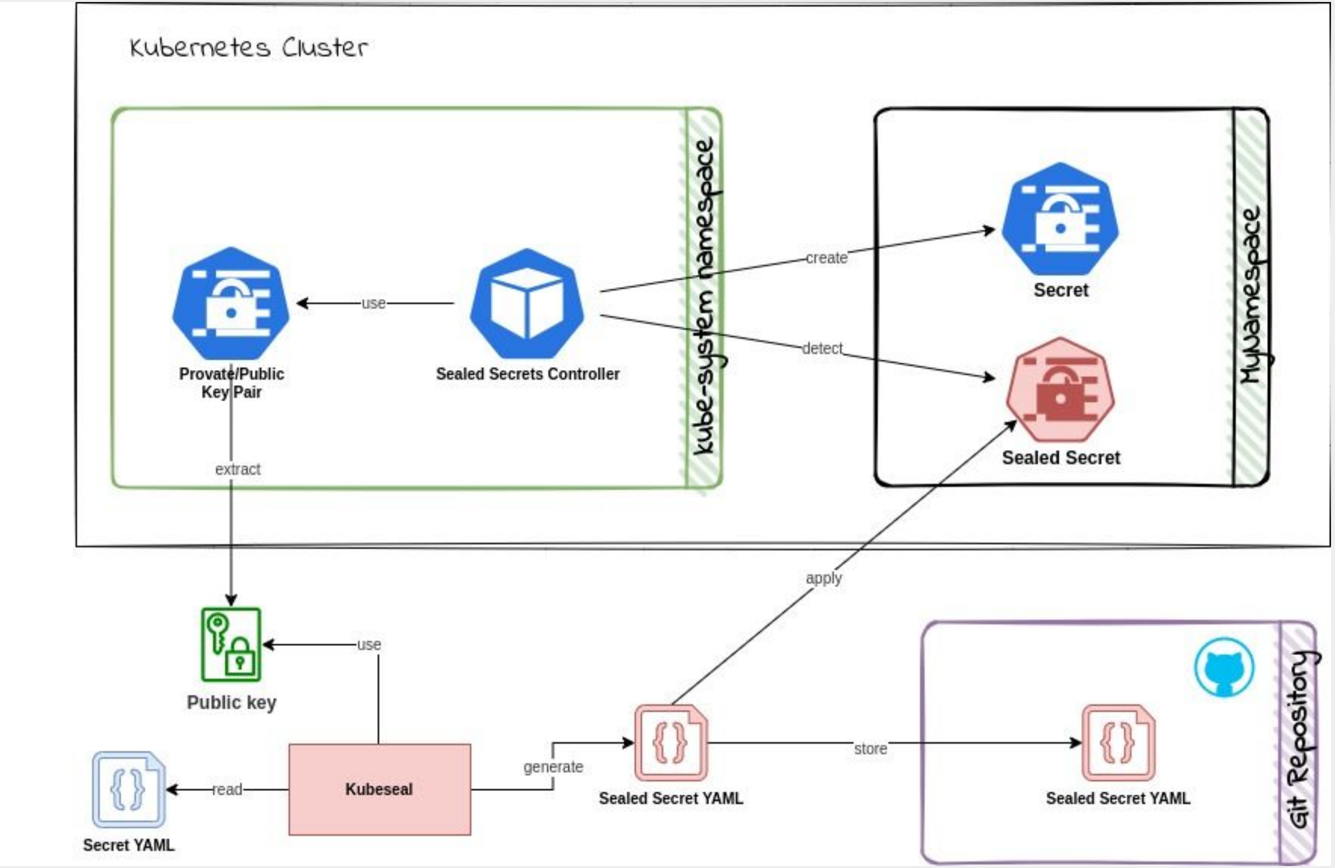
Why?

- Kubernetes Secrets are not very secure as its just base64 encoded strings
- Kubernetes Secrets cannot be stored in version control
- Kubernetes Secrets does not work with external secret managers

How?

- Use Sealed Secrets
  - Uses asymmetric crypto encryption and supports certificate rotation
  - Can be stored in version control
  - Encrypted using unique key per cluster, namespace and secret
  - Can manage existing secrets
  - Ideal for small teams

How?

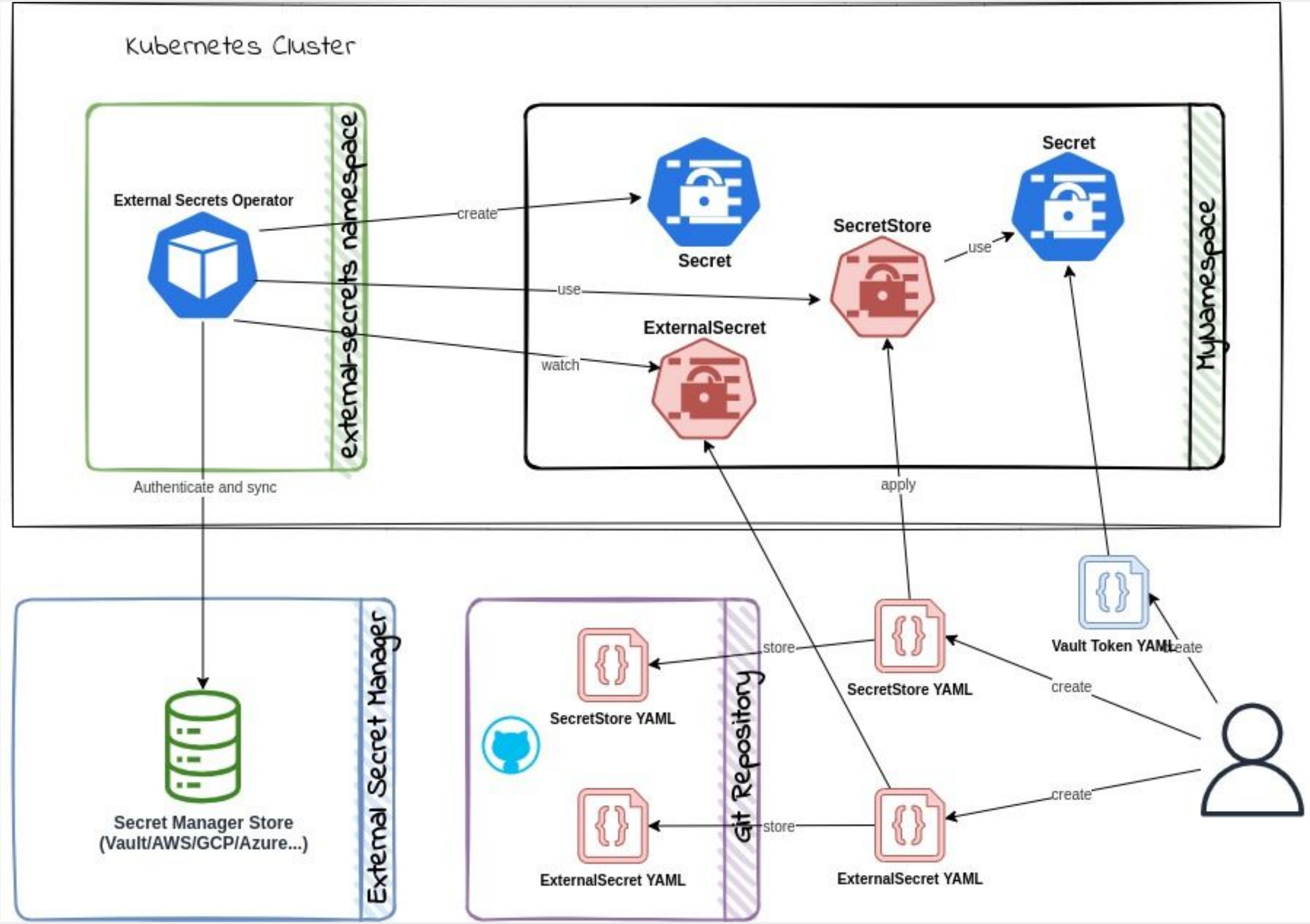


Kubernetes Cluster
kube-system namespace
MyNamespace
Provate/Public Key Pair
use
Sealed Secrets Controller
create
detect
Secret
Sealed Secret
extract
apply
Public key
use
Secret YAML
read
Kubeseal
generate
Sealed Secret YAML
store
Sealed Secret YAML
Git Repository

How?

- Use External Secrets Operator
  - Secrets are stored in external secret managers and is much more secure
  - Secrets are kept in sync
  - Works with HashiCorp Vault, Google Secrets Manager, AWS Secrets Manager and so on

How?

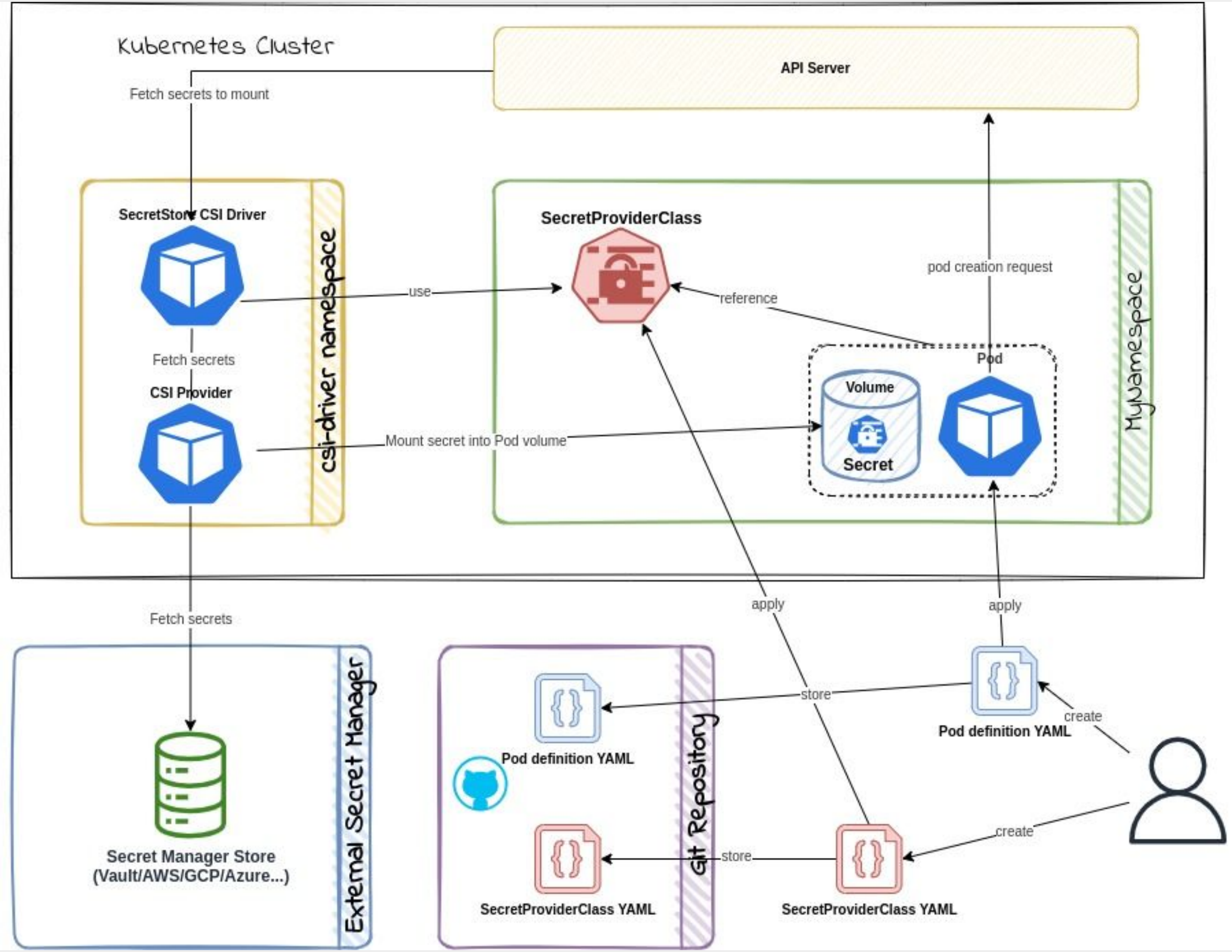


Kubernetes Cluster
External Secrets Operator
external-secrets namespace
create
use
watch
Secret
SecretStore
ExternalSecret
Secret
use
MyNamespace
Authenticate and sync
apply
External Secret Manager
Secret Manager Store
(Vault/AWS/GCP/Azure...)
SecretStore YAML
ExternalSecret YAML
Git Repository
store
store
SecretStore YAML
ExternalSecret YAML
Vault Token YAML
create
create
create

How?

- Use Secrets Store CSI driver
  - Secrets are stored in external secret managers and is much more secure
  - Secrets are mounted as volume on the pod
  - Secrets are kept in sync
  - Supports secret rotation
  - Works with HashiCorp Vault, Google Secrets Manager, AWS Secrets Manager and so on

How?



Kubernetes Cluster
API Server
Fetch secrets to mount
SecretStore CSI Driver
csi-driver namespace
use
SecretProviderClass
reference
pod creation request
MyNamespace
Pod
Volume
Secret
Fetch secrets
CSI Provider
Mount secret into Pod volume
apply
apply
Fetch secrets
External Secret Manager
Secret Manager Store
(Vault/AWS/GCP/Azure...)
Git Repository
Pod definition YAML
store
Pod definition YAML
create
SecretProviderClass YAML
store
SecretProviderClass YAML
create

How?

Shhhh... Kubernetes Secrets Are Not Really Secret!

https://auth0.com/blog/kubernetes-secrets-management/

Keep Kubernetes version up to date

Why?

- Fix CVEs and other security bugs
- Latest features and security updates

How?

- Check the Kubernetes security and disclosure information website to see if there are known security vulnerabilities for your version
- If you are using a managed PaaS, upgrade using built-in mechanism
- For on-prem installations, use tools like kOps, kubeadm, and so on, for easy upgrades

Restrict kubelet, API, and SSH access

Why?

- Restrict unintended access
- Non-admin users should not have API, SSH access

How?

- Secure API server using OIDC and RBAC
- Disable SSH for non-admin users
- Secure kubelet's HTTP endpoints

Control traffic between pods and clusters

Why?

- A compromised pod could compromise another leading to a chain reaction
- Larger attack surface
- Better traffic control and better security

How?

- Use Kubernetes network policies to control traffic between pods and clusters
- Allow only necessary traffic between pods

Use namespaces to isolate workloads

Why?

- Isolating workloads in namespaces reduces attack surface
- Easier to manage with RBAC

How?

- Avoid using default namespace
- Tune RBAC to restrict access to only required namespaces
- Use Kubernetes network policies to control traffic between namespaces

Limit resource usages

Why?

- Avoid denial of service (DoS) attacks
- Reduce attack surface

How?

- Use resources quotas and limit ranges to set limits at the namespace level
- Set resource limits at container level as well

Use monitoring tools and enable audit logging

Why?

- Detect unauthorized access attempts
- Keep an eye on the traffic
- Prevent breaches before with alarms

How?

- Enable audit logging for the cluster
- Use a monitoring tool to monitor ingress/egress networking traffic

Infrastructure best practices

How?

- Ensure that all communication is done via **TLS**.
- Protect **etcd** with TLS, Firewall, and Encryption and restrict access to it using strong credentials.
- Set up **IAM** access policies in a supported environment like a PaaS.
- Secure the Kubernetes Control Plane.
- Rotate infrastructure credentials frequently.
- Restrict cloud metadata API access when running in a PaaS like AWS, Azure, or GCP.

Container Best practices

https://a0.to/container-security

Do not run containers as root

Why?

- Principle of least privilege to reduce attack surface
- Avoid container escape and privilege escalations

How?

- Use a least privileged user
- Use `--chown=user:user` when using Docker copy commands

Use minimal up-to-date official base images

Why?

- Reduce attack surface
- Latest bug fixes and security patches

How?

- Use deterministic image tags - `FROM node:14.2.0-alpine3.11` instead of `FROM node:14-alpine`
- Install only production dependencies
- Use [official verified images]() for popular software. Prefer LTS versions.
- Use a trusted registry for non-official images and always verify the image publisher

Prevent loading unwanted kernel modules

Why?

- Reduce attack surface
- Better performance

How?

- Restrict using rules in /etc/modprobe.d/kubernetes-blacklist.conf of the node
- Uninstall the unwanted modules from the node

Enable container image scanning in your CI/CD phase

Why?

- Detect known vulnerabilities before they are exploited

How?

- Enable image scanning in CI/CD phase
- Use OSS tools like clair, Anchore or commercial tools like Snyk

Audit images

Why?

- Check for security best practices

How?

- Use Docker Bench for Security to audit your container images

Use pod security policies

Why?

- Reduce attack surface
- Prevent privilege escalation

How?

- Use Pod Security Admission to limit a container's access to the host further

Thank you!

Deepu K Sasidharan

@deepu105@bsky.social | deepu.tech