# Kubernetes风险检测技术

安易科技 王亮

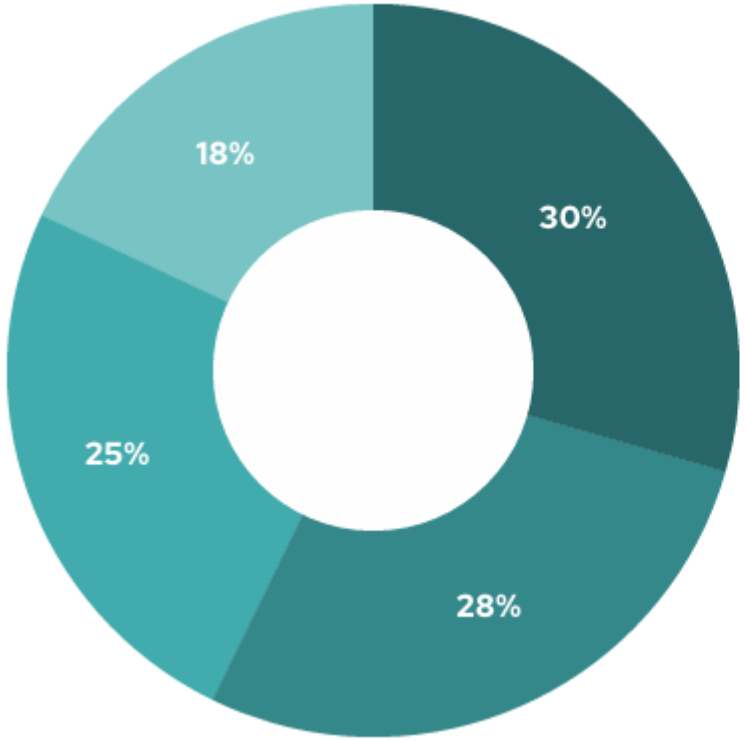
# 目录

CONTENTS

# 1 Kubernetes两类风险

# 配置错误和漏洞是影响Kubernetes最大的风险






**30%**
Vulnerabilities

**28%**
Misconfigurations/exposures

**25%**
Attacks

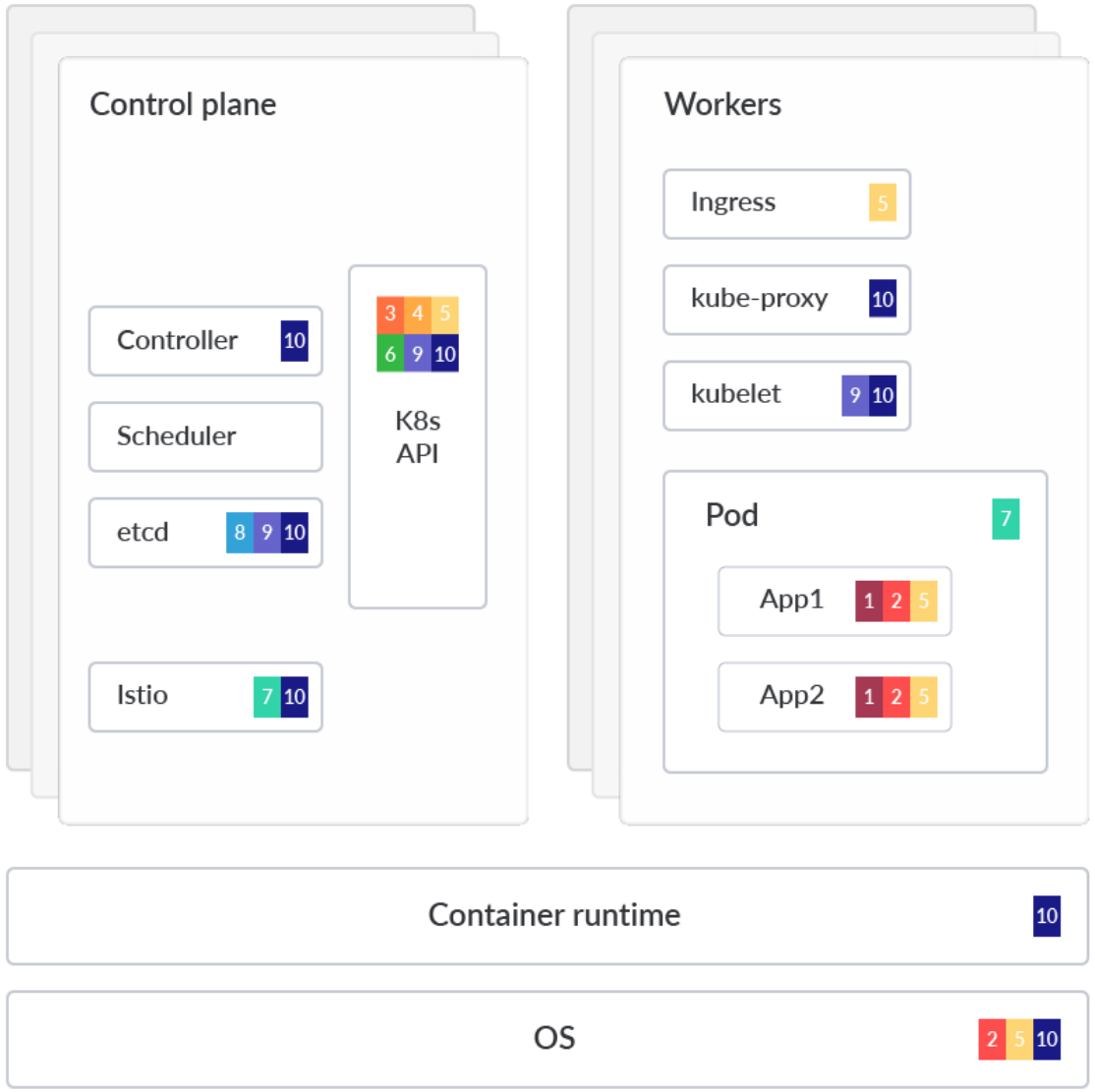
**18%**
Failing compliance (SOC2, PCI, HIPAA, etc.)

在Red Hat 2023 年 Kubernetes 安全状况报告中，超过 50% 的受访者表示他们担心配置错误和漏洞。

- 随着资源之间的差距不断扩大和软件漏洞的增加，拥有一种预防策略至关重要。采取预防措施比对每个报告的配置错误或常见漏洞和曝光（CVE）做出反应更有效。
- 一个坏的行为者只需要一个小小的配置错误就能造成严重破坏。
- 容器镜像中的现有漏洞或恶意软件也构成了重大风险。虽然容器镜像在 Kubernetes 部署中起着关键作用，但利用过时或有漏洞的镜像会引入安全风险。恶意行为者可以针对容器镜像中已知的漏洞，获取未经授权的访问权限或执行恶意代码。

# Kubernetes影响最大的10类风险最大的风险






配置错误：

- 工作负载配置不当
- 集群组件配置不当
- RBAC配置不当
- 缺失网络隔离控制
- 缺失日志和监控
- 缺失集中策略执行
- 缺失机密管理
- 身份验证机制不健全

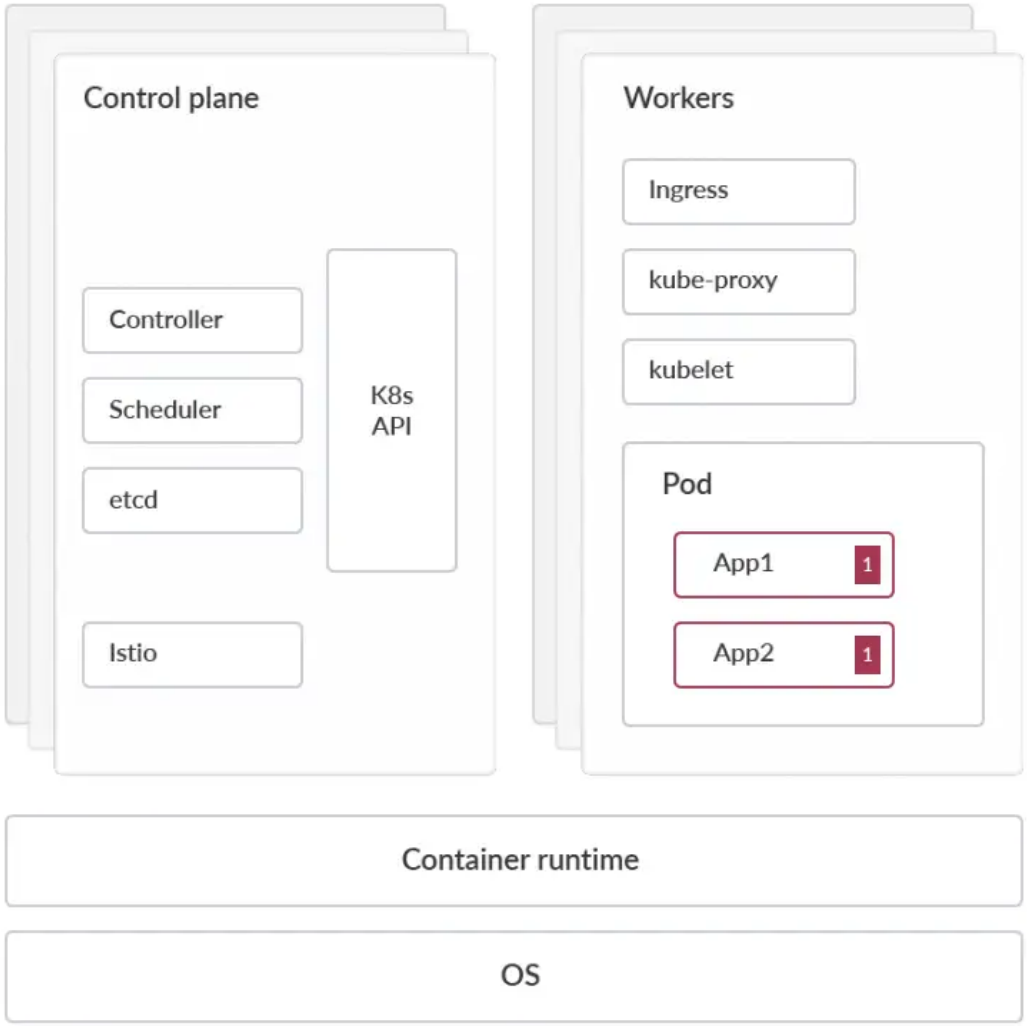
漏洞：

- 供应链漏洞
- Kubernetes组件漏洞

# 2 Kubernetes风险详述

# 不安全的工作负责配置风险






Container registry

1 K01

**Insecure Workloads Configurations**

- App processes should not run as root
- Read-only filesystems should be used
- Privileged containers should be disallowed

- **审核工作负载**

最小化 root 容器的准入,建议所有容器应作为定义的非 UID 0 用户运行。审核yaml文件。

- **使用 OPA 防止工作负载配置错误**

在特权模式下运行 pod 意味着该 pod 可以访问主机的资源和内核功能。为了防止特权 pod，OPA Gatekeeper 准入控制器的 .rego 文件类似于:

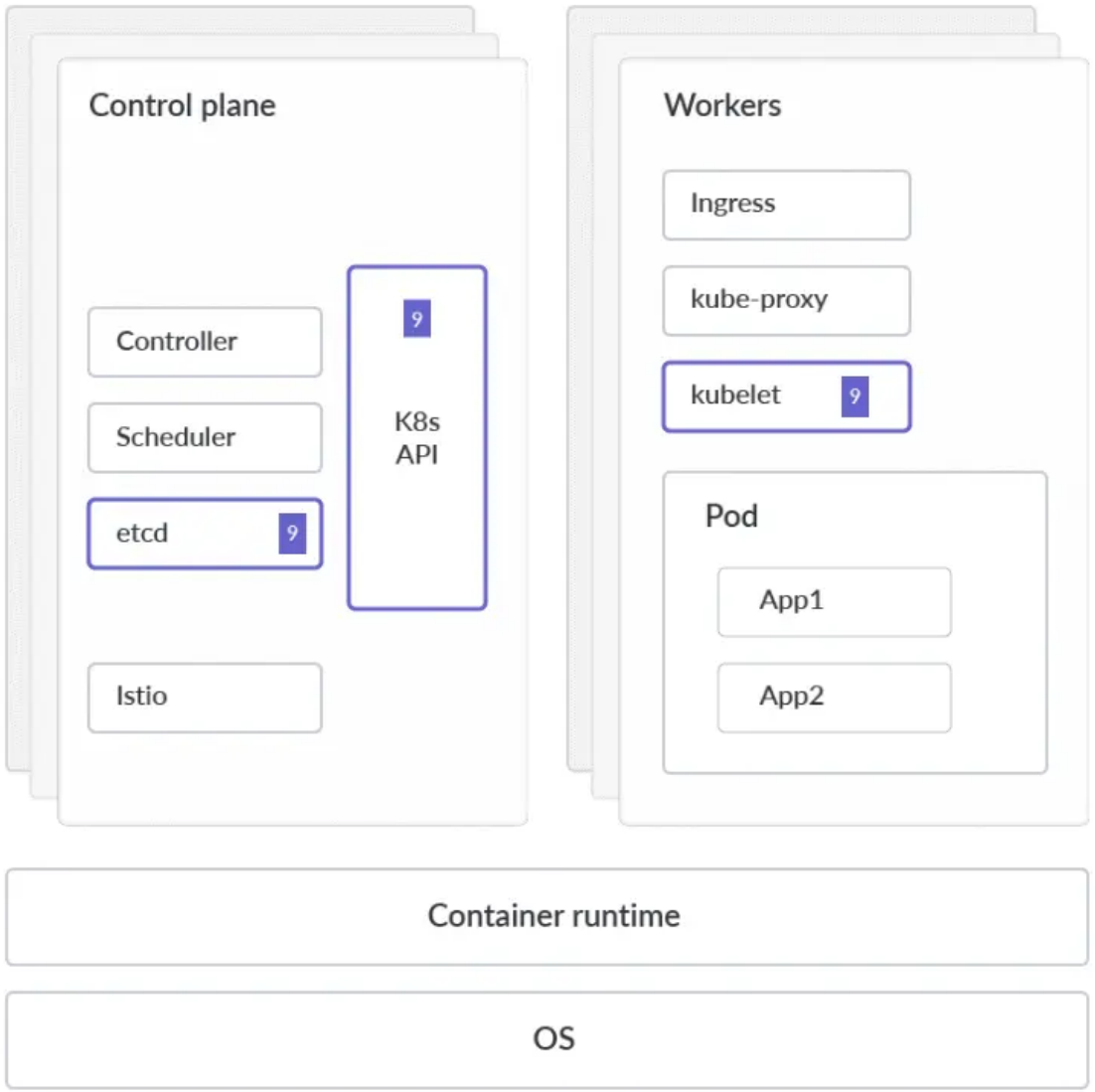
```
packagekubernetes.admission
deny[msg]{ c:=input.containers[_]
c.securityContext.privileged
msg:=sprintf("Privilegedcontainerisnotallowed:%v,securityContext:%v",[c.name,c.securityContext]) }
```

输出应该类似于以下内容:

```
Errorfromserver(Privilegedcontainerisnotallowed:alpine,securityContext:{"privileged":true}):errorwhencreating"STDIN":admissionwebhook"validating-webhook.openpolicyagent.org"
```

# 集群组件配置错误风险






Container registry

K09

Misconfigured Cluster Components

- kubelet
- etcd
- kube-apiserver

- **Kubelet 中的匿名认证设置**

kubelet 的 HTTPS 端点的所有请求，如果没有被其他配置的身份验证方法拒绝，则被视为匿名请求，并被赋予用户名 system:anonymous 和组 system:unauthenticated。启动 kubelet 时使用特性标志 --anonymous-auth=false。Kube-bench

  - POST
  - GET
  - PUT
  - PATCH
  - DELETE

- **etcd**

使用 etcdctl snapshot save 命令从活动集群成员中获取快照。将 keyspace 从 $ENDPOINT 服务复制到文件 snapshotdb 中获取快照的示例：

ETCDCTL_API=3etcdctl--endpoints$ENDPOINTsnapshotsavesnapshotdb

- **kube-apiserver**

在 kube-apiserver 中使用特性标志--tls-cert-file=[file]和--tls-private-key-file=[file]启用 TLS。

- **CoreDNS**

对 CoreDNS 的合规性检查

# 不恰当的RBAC访问控制策略风险



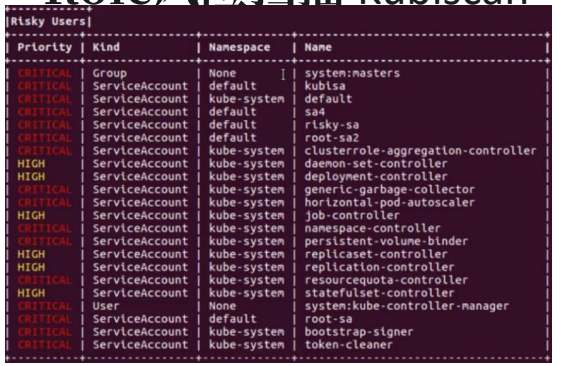


3 K03
Overly Permissive RBAC

- **ClusterRole资源限制**

阻止对 pods 进行创建、更新或删除操作，策略会类似于以下示例：
apiVersion:rbac.authorization.k8s.io/v1
kind:Role Metadata: namespace:default
name:pod-reader Rules: -apiGroups:[""]#""表示核心API组 resources:["pods"]
verbs:["get","watch","list"]
Role-Binding
apiVersion:rbac.authorization.k8s.io/v1
kind:RoleBinding Metadata: name:read-pods
namespace:default Subjects: -kind:User
name:nigeldouglas
apiGroup:rbac.authorization.k8s.io roleRef: kind:Role
name:pod-reader apiGroup:rbac.authorization.k8s.io

- **Role风险扫描** Kubiscan

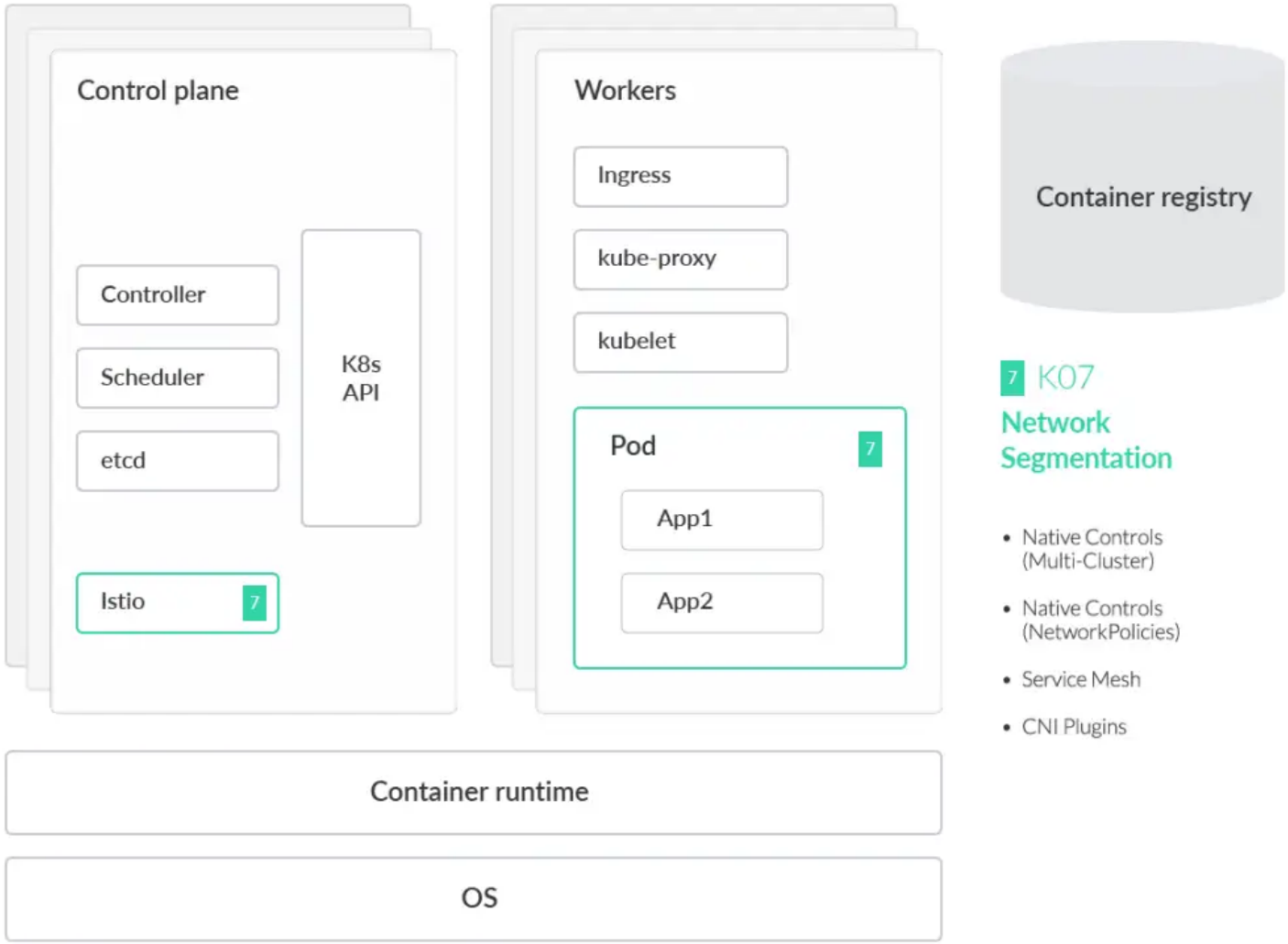


Risky Users

| Priority | Kind | Namespace | Name |
|---|---|---|---|
| CRITICAL | Group | None | system:masters |
| CRITICAL | ServiceAccount | default | kubisa |
| CRITICAL | ServiceAccount | kube-system | default |
| CRITICAL | ServiceAccount | default | sa4 |
| CRITICAL | ServiceAccount | default | risky-sa |
| CRITICAL | ServiceAccount | default | root-sa2 |
| CRITICAL | ServiceAccount | kube-system | clusterrole-aggregation-controller |
| HIGH | ServiceAccount | kube-system | daemon-set-controller |
| HIGH | ServiceAccount | kube-system | deployment-controller |
| CRITICAL | ServiceAccount | kube-system | generic-garbage-collector |
| CRITICAL | ServiceAccount | kube-system | horizontal-pod-autoscaler |
| HIGH | ServiceAccount | kube-system | job-controller |
| CRITICAL | ServiceAccount | kube-system | namespace-controller |
| CRITICAL | ServiceAccount | kube-system | persistent-volume-binder |
| HIGH | ServiceAccount | kube-system | replicaset-controller |
| HIGH | ServiceAccount | kube-system | replication-controller |
| CRITICAL | ServiceAccount | kube-system | resourcequota-controller |
| HIGH | ServiceAccount | kube-system | statefulset-controller |
| CRITICAL | User | None | system:kube-controller-manager |
| CRITICAL | ServiceAccount | default | root-sa |
| CRITICAL | ServiceAccount | kube-system | bootstrap-signer |
| CRITICAL | ServiceAccount | kube-system | token-cleaner |

# 缺少网络隔离控制风险






- **Istio Authorization Policy**

处理集群内服务之间的入口/出口流量，以及从服务到服务网格架构中的外部服务的流量：

- **CNI**

关注网络层流量（L3/L4）**NetworkPolicy**

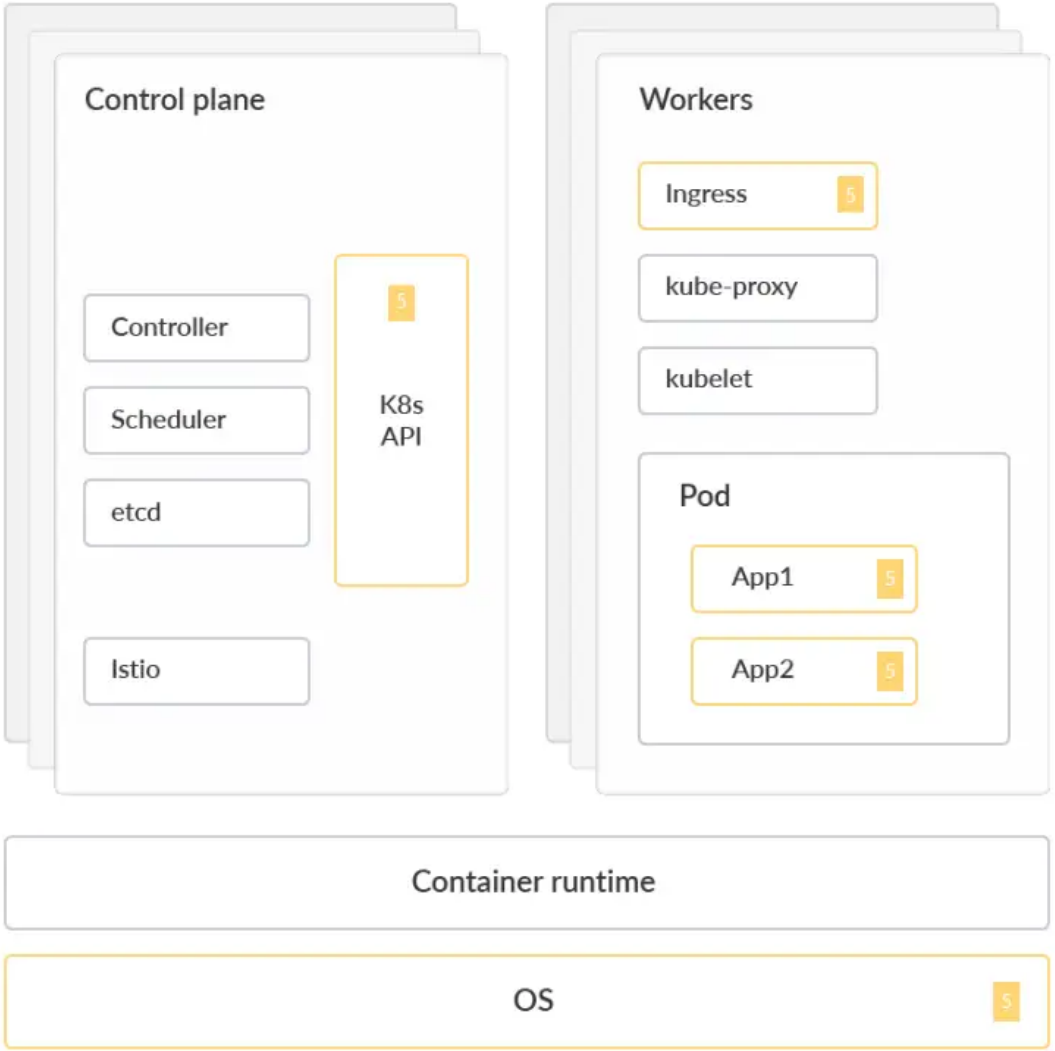
示例：

将所有带有标签 app=frontend 的端点限制为仅能够向 TCP:80 目标地址发送数据包：

apiVersion:"cilium.io/v2" kind:CiliumNetworkPolicy Metadata: name:"l4-rule" Spec: endpointSelector: matchLabels: app:frontend Egress: -toPorts: -ports: -port:"80" protocol:TCP

# 日志和监控不足风险






Container registry

K05

Inadequate Logging

- Kubernetes Events
- Application & Container Logs
- Operating System Logs
- Cloud Provider Logs
- Network Logs

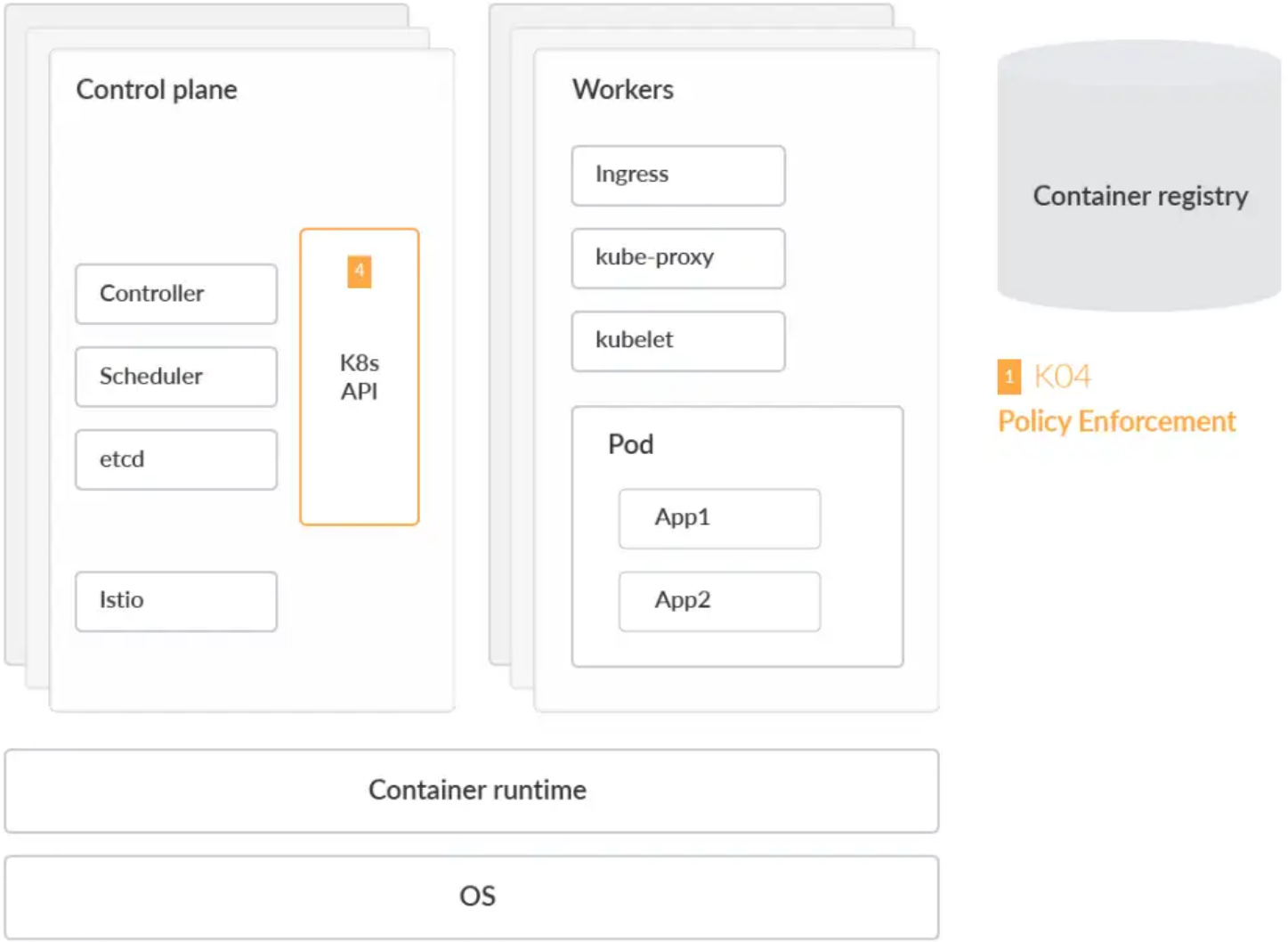
- **使用第三方软件监控**
  - Prometheus
  - Grafana
  - Falco
- **Kube-API日志**

请求和响应的日志记录：

  - apiVersion:"cilium.io/v2" •创建和销毁pod、服务、部署、守护进程等。
  - 创建、更新和删除 ConfigMap 或 secrets。
  - 订阅对任何端点所做的更改。

# 缺乏集中策略执行风险






**准入控制器 OPA**

API请求必须先经过身份验证和授权
apiVersion:apiserver.config.k8s.io/v1
kind:AdmissionConfiguration plugins: -
name:ImagePolicyWebhook configuration:
imagePolicy: kubeConfigFile:<path-to-kubeconfig-file> allowTTL:50 denyTTL:50 retryBackoff:500 defaultAllow:true

**集中管理策略**

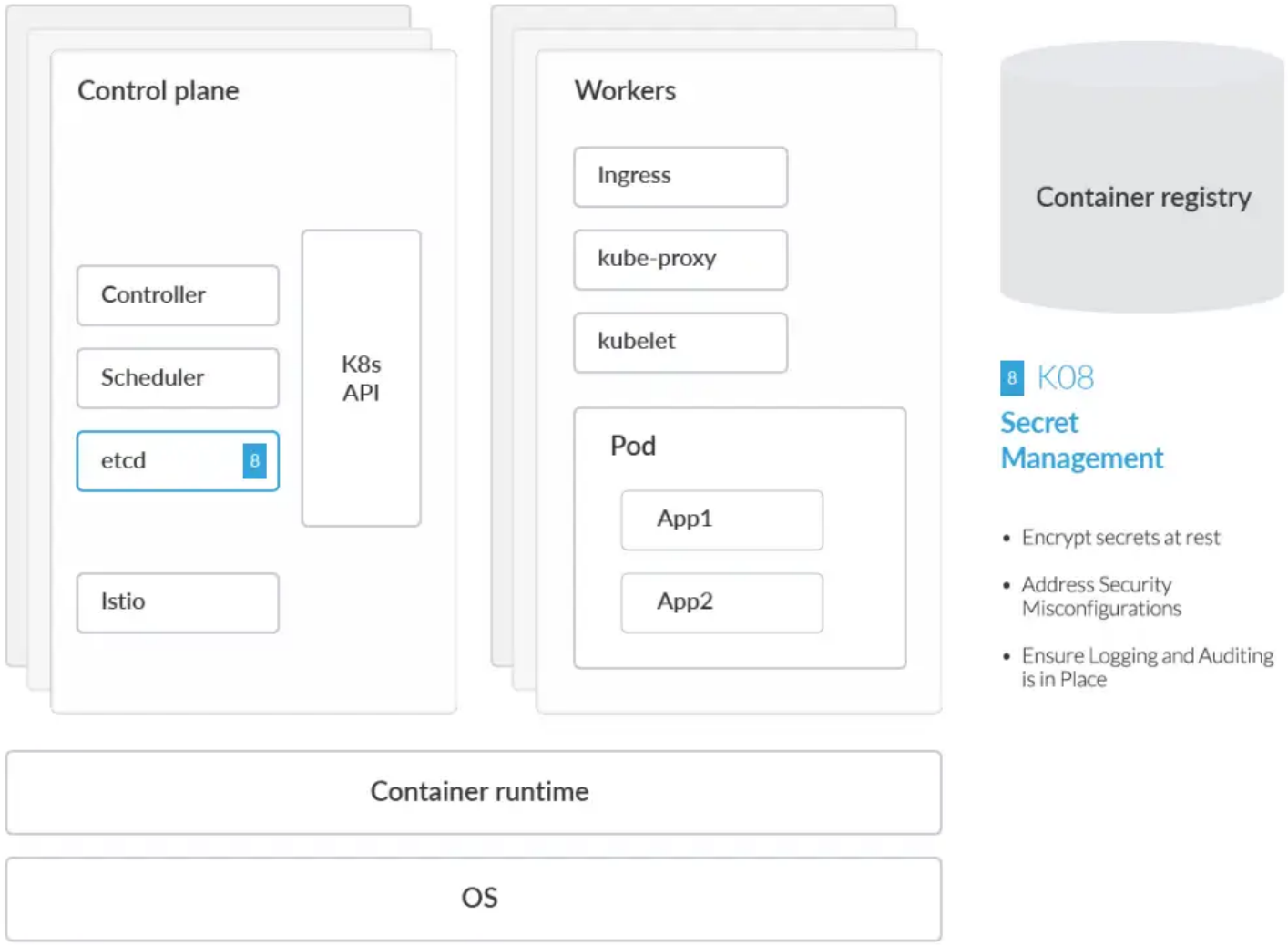
kube-mgmt跨集群使用特性标志--enable-policy=false禁用策略，或者同样可以通过单个标志禁用数据：--enable-data=false

**运行时检测：**ConfigMaps 中暴露的私密凭据

condition:(ka.req.configmap.objcontains"aws_access_key_id“ or ka.req.configmap.objcontains“ aws-access-key-id“ or ka.req.configmap.objcontains“ aws_s3_access_key_id“ or ka.req.configmap.objcontains"aws-s3-access-key-id"or ka.req.configmap.objcontains"password"or ka.req.configmap.objcontains"passphrase")

# 机密管理失败风险

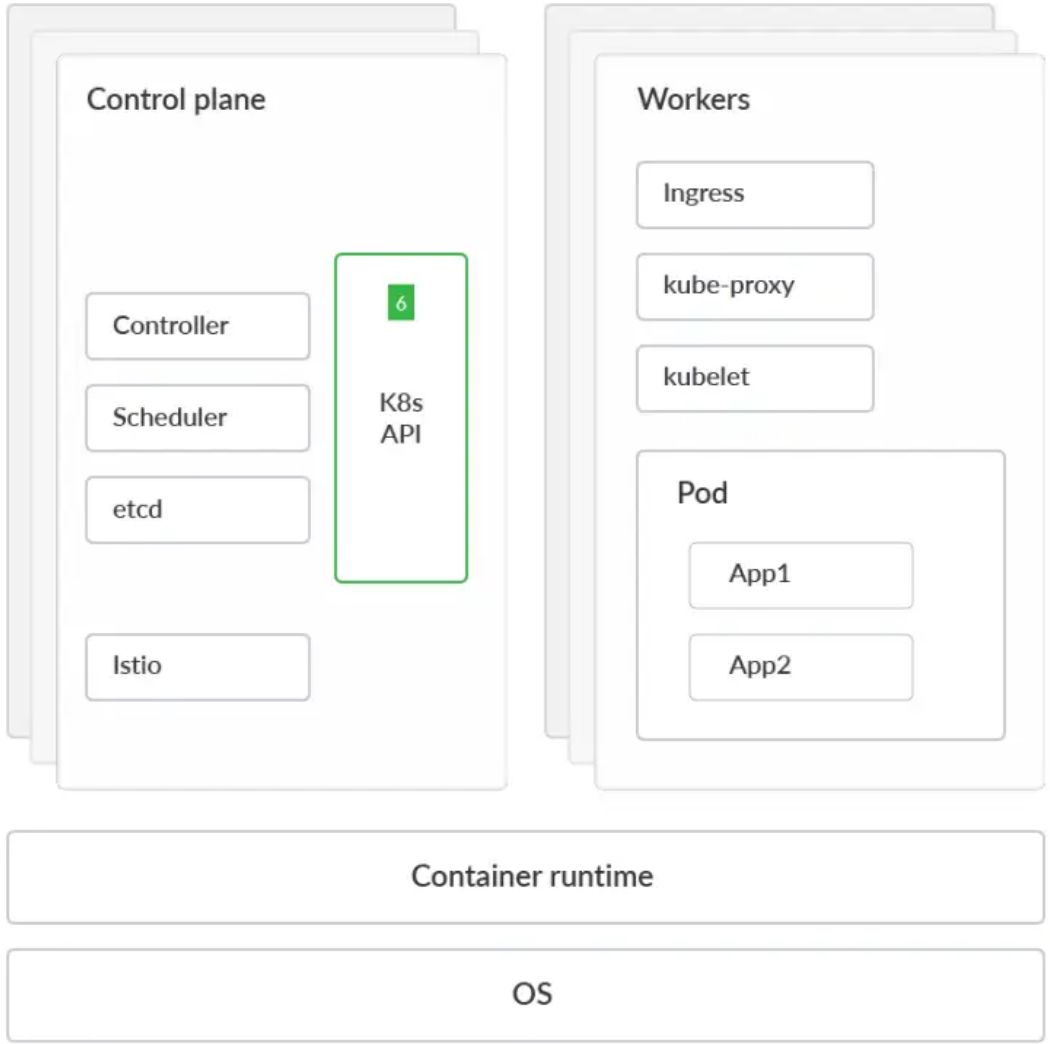





- **在 etcd 中加密 Secret 资源**
  创建 EncryptionConfiguration 自定义资源的示例：
  apiVersion:apiserver.config.k8s.io/v1 kind:EncryptionConfiguration resources: -resources: -secrets providers: -aescbc: Keys: -name:key1 secret:<BASE64ENCODEDSECRET> -identity:{}
- **解决安全配置错误**
  所有服务账户和用户访问权限保持在最小特权
  每个用户都应该有明确定义的服务账户名称不要使用admin、default。
  apiVersion:rbac.authorization.k8s.io/v1 kind:ClusterRole Metadata: name:secret-reader namespace:test Rules: -apiGroups:[""] resources:["secrets"] verbs:["get","watch","list"]
- **日志记录和审计：**写入标准输出（stdout）和标准错误流

# 不健全的认证风险






Container registry

6 K06

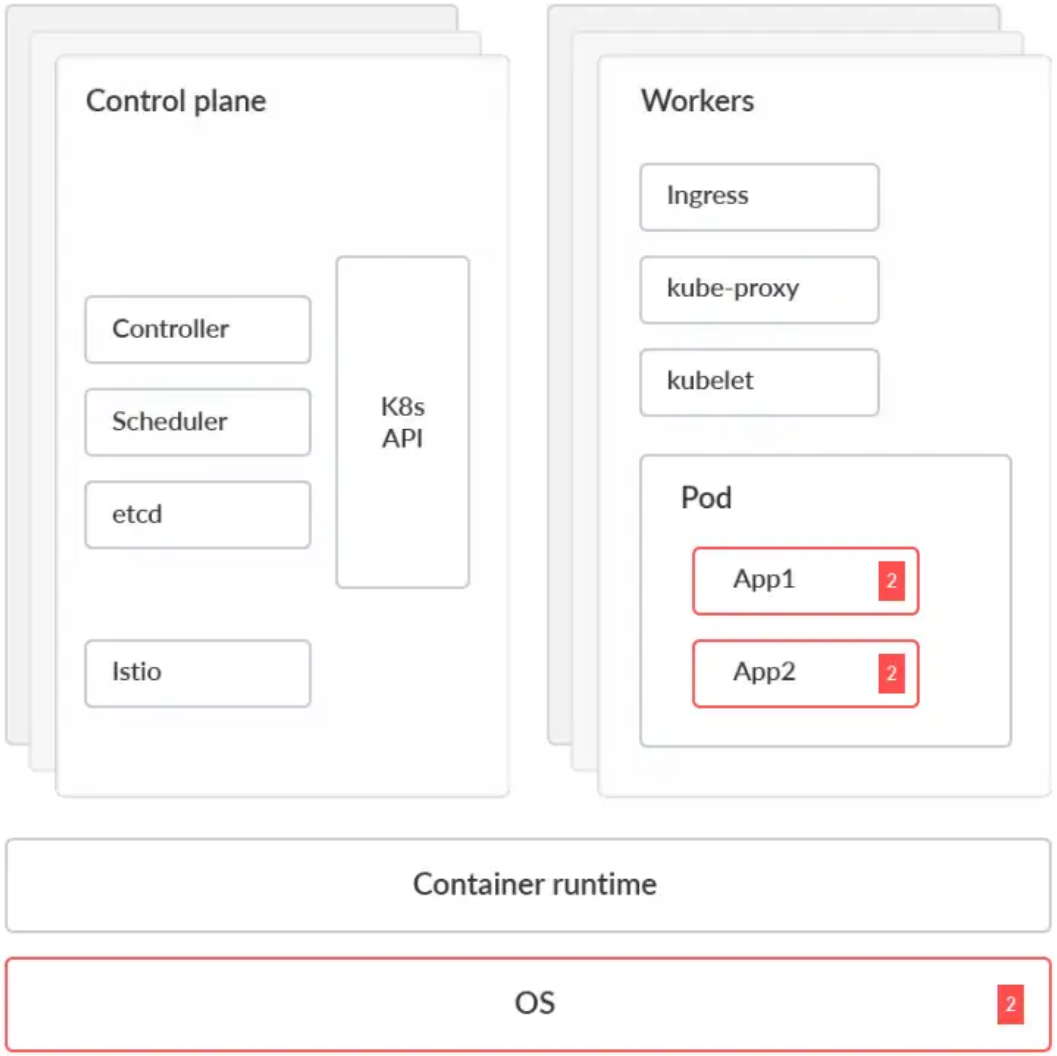
Broken Authentication

- Avoid using certificates for end-user authentication
- Never roll your own authentication
- Enforce MFA when possible
- Don't use Service Account tokens from outside of the cluster
- Authenticate users and external services using short-lived tokens

- **Kubernetes认证弱点**
  - 仪表盘
  - API
- 2FA
  - 双因素认证身份。

# 软件供应链漏洞风险







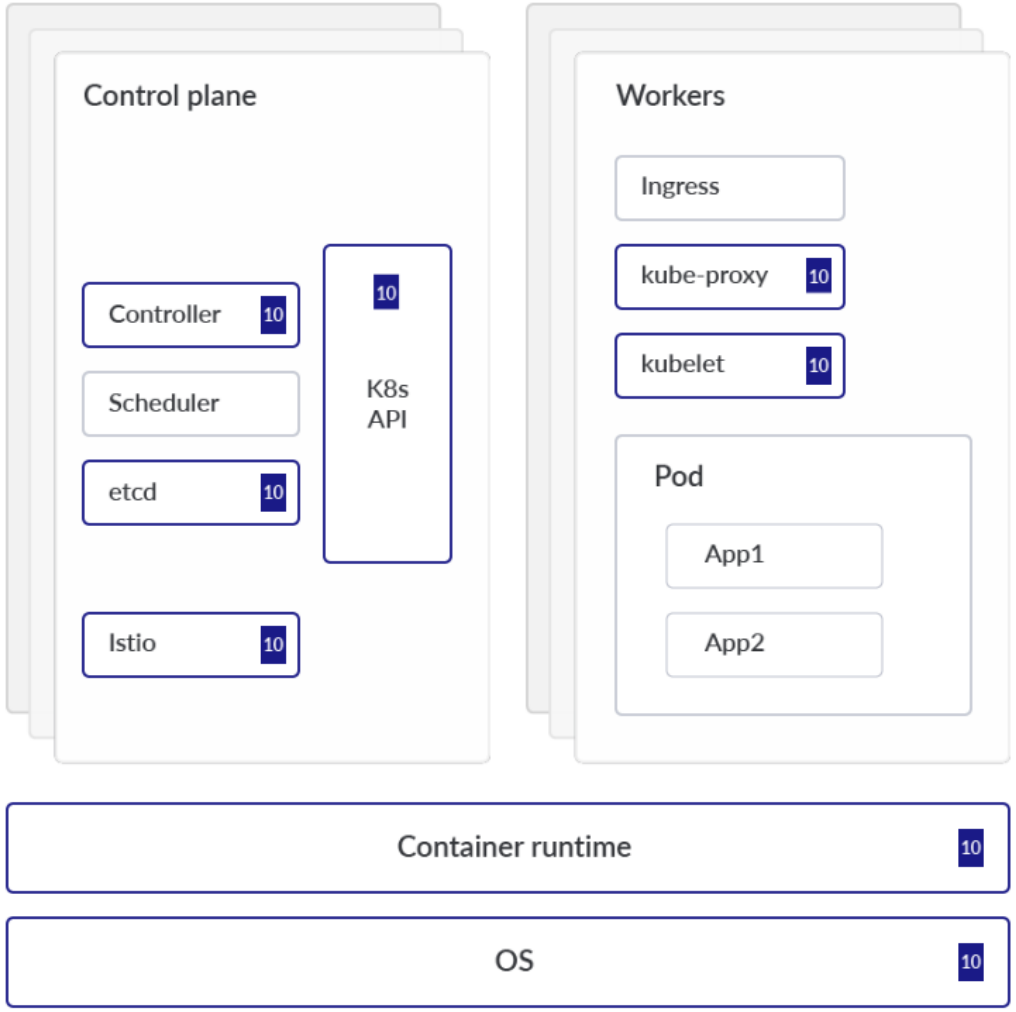
K02
Supply Chain Vulnerabilities

- Image Integrity
- Image Composition
- Known Software Vulnerabilities

- **镜像漏洞**
  - Crypto mining
  - Embedded secrets
  - Proxy avoidance
  - New registered domains
  - Malicious websites
  - Hacking
  - Dynamic DNS
- **依赖关系 Kubeview**
  可视化显示部署、服务、持久卷索赔（PVC）等之间依赖关系。

# K8S组件漏洞风险







10 K10

**Vulnerable K8s Components**

- kubelet
- etcd
- kube-apiserver

- **K8S组件漏洞 CVE Feed**
  - CVE-2021-25735
  - CVE-2020-8554
  - CVE-2019-11246
  - CVE-2018-18264
  - 通过 Kubernetes 仪表板进行特权升级
- **Linux Kernel漏洞 Falco**
- **容器运行时漏洞 Falco**

CSA GCR cloud security alliance GREATER CHINA REGION

TCCS

NSFOCUS 绿盟科技

安易科技 ANESEC TECHNOLOGY

# THANKS