# Q/KXY

云 计 算 开 源 产 业 联 盟 技 术 文 件

Q/KXY CS001—2023

# 云原生安全配置基线规范

Cloud-native Security Configuration Baseline Specification



**2023-07-25 发布** **2023-07-28 实施**

云计算开源产业联盟

# 目　　次

# 前　言

本文件按照GB/T 1.1—2020给出的规则起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本文件由云计算开源产业联盟提出并归口。

本文件起草单位：中国信息通信研究院、华为云计算技术有限公司、阿里云计算有限公司、腾讯云计算（北京）有限责任公司、天翼云科技有限公司、中国移动信息技术有限公司、中移（苏州）软件技术有限公司、兴业数字金融服务（上海）股份有限公司、北京银行股份有限公司、北京小佑网络科技有限公司、绿盟科技集团股份有限公司、北京升鑫网络科技有限公司、奇安信科技集团股份有限公司、杭州安恒信息技术股份有限公司、厦门服云信息科技有限公司、北京探真科技有限公司、杭州默安科技有限公司、安易科技（北京）有限公司、北京东方通科技股份有限公司、北京华云安信息技术有限公司、安超云软件有限公司、北京启明星辰信息安全技术有限公司、新华三技术有限公司

本文件主要起草人：栗蔚、陈屹力、刘如明、杜岚、仇保琪、刘方外、赵华、张琦、刘建峰、郑志强、匡大虎、张恒、宋志明、刘斌、王浩硕、张潜、郭旸、谢瑒、梅洪彰、王鑫、王子健、白黎明、黄竹刚、张婉莹、张政、杨冬富、张万兴、顾欢、胡俊、唐耀华、何正民、马志伟、胡向亮、左伟震、林明峰、陈俊杰、刘拓、郭嘉伟、凌惜沫、王亮、林洁、王阔阔、马维士、付建邦、隋成龙、许刚、郑斌、郭伟华、赵梓渊、蒋振超。

# 云原生安全配置基线规范

## 1 范围

本文件规定了云原生安全配置基线扫描应具备的基础规范要求。云原生安全配置基线扫描规范要求包括API Server安全配置要求、控制管理器安全配置要求、调度器安全配置要求、工作负载安全配置要求、etcd安全配置要求、kube-proxy安全配置要求、kubelet安全配置要求、CNI和网络策略安全配置要求。

本文件适用于指导云原生安全Kubernetes配置基线产品建设和相关云原生安全产品基线扫描能力建设。本文件同时适用于公有云、私有云和混合云服务及软件产品，依据交付形式的差异，本标准针对不同使用场景其技术指标要求也有所不同。

## 2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 25069-2022 信息安全技术 术语

GB/T 18336.1-2015 信息技术 安全技术 信息技术安全评估准则 第1部分：简介和一般模型

## 3 术语、定义和缩略语

### 3.1 术语和定义

GB/T 25069-2022、GB/T 18336.1-2015界定的及下列术语和定义适用于本文件。

**基线控制** baseline controls

为某一系统或组织建立的最低防护措施的集合。

[来源：GB/T 25069-2022，定义 3.262]

**配置管理** configuration management;CM

应用技术和行政指导及监督的如下行为准则：识别和记录配置项的功能和物理特性，控制对这些特性的变更，记录和报告变更处理和实施状态，并验证是否符合规定的要求。

［来源：GB/T 18336.1-2015,3.4.3，有修改：删除注等］

### 3.2 缩略语

下列缩略语适用于本文件。

API：应用程序编程接口。

## 4 框架概述

本文件中的云原生安全配置基线要求是对云原生工具Kubernetes的安全配置基线的要求，如图1所示，本文件中的要求主要包括对Kubernetes主节点API Server、控制管理器、调度器、etcd的安全配置要求，对工作节点kube-proxy、kubelet、工作负载的安全配置要求，以及针对附加项的CNI和网络策略配置的要求。

除了对应的安全要求之外，为了配合自动化基线扫描工具的使用，本文件中的各项要求均附有自动化要求。自动化要求是基于工具对基线具有自动检测的能力，并可以验证为通过/失败状态，但是检测结果是否被采纳并根据检测结果进行架构，不在自动化能力表述的范围内，规范中不能自动化的要求是表示不能对其进行技术检测的完全自动化并输出验证结果，一般需要手动步骤来验证配置的状态是否按预期设置，预期状态可能根据环境而异。


图1 云原生安全配置基线要求框架

本文件第5章为对API Server安全配置、控制管理器安全配置、调度器安全配置、etcd安全配置、kube-proxy安全配置要求、kubelet安全配置、工作负载安全配置、CNI和网络策略安全配置的具体要求，附录A、附录B为资料性附录，附录A附有第5章中各项配置要求的检查方法和恢复方法，附录B附有部分要求的典型场景。

## 5 云原生安全配置基线要求

### 5.1 API Server 安全配置要求

#### 5.1.1.文件目录安全

apiserver的pod描述文件以及相应的PKI目录和文件，包含了组件的启动参数和证书文件，这些都是直接配置apiserver的关键文件。在创建集群的过程中，这些文件会被保留在集群中，用于后续的配置修改。鉴于这些文件的重要性，因此必须确保它们的完整性和机密性，防止它们被任意或恶意篡改。因此，对这些文件的所有权和权限的管理需要严格执行。具体要求见下：

表 1

| 序号 | 对象 | 对象涉及文件 | 对文件的要求 | 自动化 |
|---|---|---|---|---|
| 5.1.1.1 | apiserver | pod描述文件 | 权限设置应为600或更严格 | 是 |
| 5.1.1.2 | apiserver | pod描述文件 | 文件属主应为root:root | 是 |
| 5.1.1.3 | apiserver | Kubernetes PKI证书文件 | 权限应设置为600或更严格 | 是 |
| 5.1.1.4 | apiserver | Kubernetes PKI证书文件 | 文件所属应设置为root:root | 是 |

### 5.1.2 身份认证和访问控制

apiserver需要进行身份认证和访问控制，其目的是为了保护集群资源的安全，其过程主要包括三个步骤：认证、授权和准入控制（主要关注资源管理）。身份认证和授权是限制访问集群资源的关键手段。如果集群配置不当，例如允许匿名访问，网络攻击者可以扫描常用的 Kubernetes 端口，无需认证就能访问集群的数据库或进行 API 调用。因此，需要合理配置apiserver的认证和鉴权设置，以在接收和发起请求时启用认证。同时，应启用适当的准入控制插件，以减少集群暴露的攻击面，从而保护集群的安全。具体要求见下：

表 2

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.1.2.1 | apiserver | --anonymous-auth | 应为false，若用户使用RBAC授权方式设定匿名用户权限进行健康检查和服务方式，通常可设置为true | 否 |
| 5.1.2.2 | apiserver | --token-auth-file | 不应设置 | 是 |
| 5.1.2.3 | apiserver | --authorization-mode | 不应设置为AlwaysAllow | 是 |
| 5.1.2.4 | apiserver | --authorization-mode | 应包含Node | 是 |
| 5.1.2.5 | apiserver | --authorization-mode | 应包含RBAC | 是 |
| 5.1.2.6 | apiserver | --service-account-lookup | 应设置为true | 是 |
| 5.1.2.7 | apiserver | --service-account-key-file | 应设置 | 是 |
| 5.1.2.8 | apiserver | --kubelet-client-certificate<br>--kubelet-client-key | 应设置 | 是 |
| 5.1.2.9 | apiserver | --kubelet-certificate-authority | 应设置 | 是 |
| 5.1.2.10 | apiserver | --tls-cert-file<br>--tls-private-key-file | 应设置 | 否 |
| 5.1.2.11 | apiserve | --client-ca-file | 应设置 | 是 |
| 5.1.2.12 | apiserver | --etcd-certfile和--etcd-keyfile | 应设置 | 是 |
| 5.1.2.13 | apiserver | --etcd-cafile | 应设置 | 是 |
| 5.1.2.14 | 准入控制插件 | EventRateLimit | 宜设置 | 否 |
| 5.1.2.15 | 准入控制插件 | AlwaysAdmit | 不应设置 | 是 |
| 5.1.2.16 | 准入控制插件 | AlwaysPullImages | 宜设置 | 否 |
| 5.1.2.17 | 准入控制插件 | SecurityContextDeny<br>(Kubernetes 1.25以下版本) | 如果PodSecurityPolicy没有设置，应确保设置该值 | 否 |
| 5.1.2.18 | 准入控制插件 | ServiceAccount | 应设置 | 是 |

表 2（续）

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.1.2.19 | 准入控制插件 | NamespaceLifecycle | 应设置 | 是 |
| 5.1.2.20 | 准入控制插件 | NodeRestriction | 应设置 | 是 |
| 5.1.2.21 | 准入控制插件 | DenyServiceExternalIPs | 不应设置- | 是 |
| 注1：AlwaysPullImages设置可能会影响脱机或孤立的集群，这些集群采用预加载镜像方式，且无法访问镜像仓库以提取使用中的镜像，此设置不适用于使用此配置的集群。对系统复位重建、堆栈升级等时长有严苛要求的业务场景，在评估安全风险的前提下，不强制要求开启AlwaysPullImages插件配置。<br>注2：针对SecurityContextDeny，Kubemetes 1.25及以上版本不做要求，但应采取等同的方式：如开启 Pod Security Standards | | | | |

### 5.1.3 防止拒绝服务攻击

请求apiserver的超时限制。默认情况下，设置为60秒，这在连接速度较慢时可能会出现问题，一旦请求的数据量超过60秒内可传输的数据量，集群资源将无法访问。但是，将此超时限制设置得太大可能会耗尽apiserver资源，使其易于遭受拒绝服务攻击。因此，建议适当设置此限制，并仅在需要时更改默认限制60秒。具体要求见下：

表 3

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.1.3.1 | apiserver | --request-timeout | 应设置 | 否 |

### 5.1.4 防止信息泄露

apiserver的不当配置会导致记录程序的详细信息（包含程序敏感数据）、使数据未加密或暴露不安全的访问端口，应该修正相关配置。具体要求见下：

表 4

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.1.4.1 | apiserver | --profiling | 应设置为false | 是 |
| 5.1.4.2 | apiserver | --experimental-encryption-provider-config<br>(Kubernetes 1.13版本及以下)<br>--encryption-provider-config<br>(Kubernetes 1.13版本以上) | 应设置 | 否 |
| 5.1.4.3 | apiserver | --secure-port | 应正确配置，需为1和65535之间的整数值。 | 是 |
| 5.1.4.4 | apiserver | --insecure-bind-address<br>(Kubernetes 1.24以下版本) | 不应设置 | 是 |
| 5.1.4.5 | apiserver | --insecure-port<br>(Kubernetes 1.24以下版本) | 应设置为0 | 是 |

### 5.1.5 日志安全

为保证对集群进行故障排查、安全监控、数据分析以及安全审计，需要合理的记录日志，保障记录的日志能够覆盖安全事件的问题，日志记录的周期足够长且能满足日志间关联分析的场景。应确保创建了审计策略，且审计策略涵盖关键的安全问题。具体要求见下：

表 5

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.1.5.1 | apiserver | --audit-log-path | 应设置 | 是 |
| 5.1.5.2 | apiserver | --audit-log-maxage | 应设置为30或合适的值 | 是 |
| 5.1.5.3 | apiserver | --audit-log-maxbackup | 应设置为10或合适的值 | 是 |
| 5.1.5.4 | apiserver | --audit-log-maxsize | 应设置为100或其它合适的值 | 是 |
| 5.1.5.5 | apiserver | --audit-policy-file | 应确保创建最小化的审计策略，Kubernetes 可以审计向 API 服务器发出的请求详情。要启用该日志记录，必须设置--audit-policy-file 参数 | 否 |

### 5.1.6 SSL/TLS 配置

apiserver 的 SSL/TLS 配置的主要目的是保证数据的安全性和完整性，以及验证通信双方的身份。具体要求见下：

表 6

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.1.6.1 | apiserver | --kubelet-https<br>(Kubernetes 1.22以下版本) | 应设置为true | 是 |
| 5.1.6.2 | apiserver | --tls-cipher-suites | 应只使用强加密套件 | 否 |
| 5.1.6.3 | apiserver | 无参数 | 人机交互场景下应禁止使用客户端证书用于身份验证 | 否 |

## 5.2 控制管理器安全配置要求

### 5.2.1 文件目录安全

控制管理器（controller manager）的 pod 描述文件包含了组件的启动参数和证书文件，是直接配置控制管理器的文件，在创建集群的时候这些文件会保留在集群中用于修改相关的配置，因此需要保证其完整性、机密性且不被随意、恶意篡改，需严格限制该文件的属主和权限，确保只有管理员，或其他具备系统管理员权限的用户才能修改该文件。具体要求见下：

表 7

| 序号 | 对象 | 对象涉及文件 | 对文件的要求 | 自动化 |
|---|---|---|---|---|
| 5.2.1.1 | controller manager | pod描述文件 | 文件权限为600或更严格 | 是 |
| 5.2.1.2 | controller manager | pod描述文件 | 文件属主应为root:root | 是 |
| 5.2.1.3 | controller manager | controller-manager.conf | 文件权限为600或更严格 | 是 |
| 5.2.1.4 | controller manager | controller-manager.conf | 文件属主应为root:root | 是 |

### 5.2.2 身份认证和访问控制

控制管理器是集群内部的管理控制中心，它的职责是保证集群中各种资源的状态和用户定义的状态一致，如果出现偏差，则修正资源的状态。如 replication controller 的核心作用是保障集群中某个 RC 关联的 pod 副本数与预设值一致。控制管理器内部包含了 node controller,namespace controller,daemonset controller 等多个控制器，每种控制器都负责一种具体的控制流程。需要确保每类控制器具备其所需的最小权限，且控制器能安全与 apiserver 建立通信。具体要求见下：

表 8

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.2.2.1 | controller manager | --use-service-account-credentials | 应设置为true | 是 |
| 5.2.2.2 | controller manager | --service-account-private-key-file | 应设置 | 是 |
| 5.2.2.3 | controller manager | --root-ca-file | 应设置 | 是 |
| 5.2.2.4 | controller manager | --RotateKubeletServerCertificate | 应设置为true | 是 |
| 注：5.2.2.4 要求仅适用于让 kubelet 从 apiserver 获取证书的情况。如果 kubelet 证书来自外部权威机构/工具，则需要自己处理轮换问题。 | | | | |

### 5.2.3 防止拒绝服务

垃圾回收对于确保足够的资源可用性和避免降低性能和可用性很重要。在最坏的情况下，系统可能崩溃或长时间无法使用导致拒绝服务。当前的垃圾收集设置是 12500 个终止的 Pod，对于系统而言，可能太高了。根据系统资源和测试，选择适当的阈值以激活垃圾收集。具体要求见下：

表 9

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.2.3.1 | controller manager | --terminated-pod-gc-threshold | 应设置 | 否 |
| 注：如果集群规模较小或资源有限，可以考虑适当降低阈值，例如设置为 5000 或更低。如果集群规模较大或资源充足，也可以稍微增加阈值，例如设置为 15000 或更高。需要注意的是，阈值的调整应该是一个渐进的过程，观察调整后的集群运行情况，并根据实际情况进行适当的调整。 | | | | |

### 5.2.4 防止信息泄露

控制管理器的不当配置会导致记录程序的详细信息（包含程序敏感数据）、使数据未加密或暴露不安全的访问端口，应该修正相关配置。具体要求见下：

表 10

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.2.4.1 | controller manager | --profiling | 应设置为false | 是 |
| 5.2.4.2 | controller manager | --bind-address | 应设置为127.0.0.1 | 是 |

## 5.3 调度器安全配置要求

### 5.3.1 文件目录安全

调度器（scheduler）的pod描述文件和配置文件，包含了组件的启动参数和证书文件，这些都是直接配置调度器的关键文件。在创建集群的过程中，这些文件会被保留在集群中，用于后续的配置修改。鉴于这些文件的重要性，需严格限制该文件的属主和权限，确保只有管理员，或其他具备系统管理员权限的用户才能修改该文件。具体要求见下：

表 11

| 序号 | 对象 | 对象涉及文件 | 对文件的要求 | 自动化 |
|---|---|---|---|---|
| 5.3.1.1 | scheduler | scheduler pod描述文件 | 文件的权限应设置为600或更严格 | 是 |
| 5.3.1.2 | scheduler | scheduler pod描述文件 | 文件属主应为root:root | 是 |
| 5.3.1.3 | scheduler | scheduler.conf文件 | 文件的权限应设置为600或更严格 | 是 |
| 5.3.1.4 | scheduler | scheduler.conf文件 | 文件属主应为root:root | 是 |

### 5.3.2 防止信息泄露

调度器的监听端口应限制在 127.0.0.1 上，且关闭调试 API，避免信息泄露。具体要求见下：

表 12

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.3.2.1 | scheduler | --profiling | 应设置为false | 是 |
| 5.3.2.2 | scheduler | --bind-address | 应设置为127.0.0.1 | 是 |

## 5.4 etcd 安全配置要求

### 5.4.1 文件目录安全

etcd 的 pod 描述文件和配置文件，以及数据文件，需严格限制该文件的属主和权限，确保只有管理员，或其他具备系统管理员权限的用户才能修改该文件。具体要求见下：

表 13

| 序号 | 对象 | 对象涉及文件 | 对文件的要求 | 自动化 |
|---|---|---|---|---|
| 5.4.1.1 | etcd | etcd pod描述文件 | 当etcd是由pod部署时，权限设置应为600或更严格 | 是 |
| 5.4.1.2 | etcd | etcd pod描述文件 | 当etcd是由pod部署时，文件属主应为root:root | 是 |
| 5.4.1.3 | etcd | etcd数据存放目录 | 权限设置应为700或更严格 | 是 |
| 5.4.1.4 | etcd | etcd数据存放目录 | 文件属主应为root:root | 是 |

### 5.4.2 身份认证和访问控制

etcd是Kubernetes部署使用的高可用性键值存储，用于持久存储其所有REST API对象。这些对象含有敏感信息，因此etcd在接受请求和发起请求时应正确配置身份认证和访问控制，具体要求见下：

表 14

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.4.2.1 | etcd | --cert-file 和 --key-file | 应设置 | 是 |
| 5.4.2.2 | etcd | --client-cert-auth | 应设置为true | 是 |
| 5.4.2.3 | etcd | --auto-tls | 应禁止设置为true | 是 |
| 5.4.2.4 | etcd | --peer-cert-file 和 --peer-key-file | 应设置 | 是 |
| 5.4.2.5 | etcd | --peer-client-cert-auth | 应设置为true | 是 |
| 5.4.2.6 | etcd | --peer-auto-tls | 应禁止设置为true | 是 |
| 5.4.2.7 | etcd | 无参数 | 宜使用独立的CA证书 | 否 |
| 注：5.4.2.4 仅适用于 etcd 集群。如果您在环境中仅使用一台 etcd 服务器，则此建议不适用。 | | | | |

### 5.4.3 SSL/TLS 配置

etcd 发起 TLS 协商应启用强加密套件，保障通信安全，具体要求见下：

表 15

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.4.3.1 | etcd | --cipher-suites | 应只使用强加密套件 | 否 |

## 5.5 kube-proxy 安全配置要求

### 5.5.1 文件目录安全

kube-proxy 的 pod 描述文件和配置文件，需要严格限制该文件的属主和权限，确保只有管理员，或其他具备系统管理员权限的用户才能修改该文件。具体要求见下：

表 16

| 序号 | 对象 | 对象涉及文件 | 对文件的要求 | 自动化 |
|---|---|---|---|---|
| 5.5.1.1 | kube-proxy | kubeconfig文件 | 权限应设置为600或更严格 | 是 |
| 5.5.1.2 | kube-proxy | kubeconfig文件 | 文件所属应设置为root:root | 是 |
| 注：若 kubeconfig 文件存储位置为默认存储位置，则可以实现自动化扫描。 | | | | |

## 5.6 Kubelet 安全配置要求

### 5.6.1 文件目录安全

kubelet 服务文件设置工作节点中 kubelet 服务行为的各种参数，这些参数设置了工作节点中 kubelet 服务的行为。需严格限制该文件的属主和权限，确保只有管理员，或其他具备系统管理员权限的用户才能修改该文件。具体要求见下：

表 17

| 序号 | 对象 | 对象涉及文件 | 对文件的要求 | 自动化 |
|---|---|---|---|---|
| 5.6.1.1 | kubelet | kubelet服务文件 | 权限设置应为600或更严格 | 是 |
| 5.6.1.2 | kubelet | kubelet服务文件 | 文件所属应设置为root:root | 是 |
| 5.6.1.3 | kubelet | kubeconfig kubelet.conf文件 | 权限设置应为600或更严格 | 是 |
| 5.6.1.4 | kubelet | kubeconfig kubelet.conf文件 | 文件所属应设置为root:root | 是 |
| 5.6.1.5 | kubelet | 客户端CA文件 | 权限设置应为600或更严格 | 是 |
| 5.6.1.6 | kubelet | 客户端CA文件 | 文件所属应设置为root:root | 是 |
| 5.6.1.7 | kubelet | kubelet的--config配置文件 | 权限设置应为600或更严格 | 是 |
| 5.6.1.8 | kubelet | kubelet的--config配置文件 | 文件所属应设置为root:root | 是 |

### 5.6.2 身份认证和访问控制

kuelet 包含客户端证书与服务端证书，客户端证书为 kubelet 用来向 apiserver 认证，表明自身身份的证书。kubelet 自身也向外提供一个 HTTPS 服务，包含若干功能特性，该场景则需要使用服务端证书。Kubelet 作为服务端接受请求和作为客户端发起请求时应进行身份认证，具体要求见下：

表 18

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.6.2.1 | kubelet | --read-only-port | 应设置为0 | 否 |
| 5.6.2.2 | kubelet | --rotate-certificates | 不应设置为false | 是 |
| 5.6.2.3 | kubelet | --tls-cert-file<br>--tls-private-key-file | 应设置为合适值 | 否 |
| 5.6.2.4 | kubelet | --client-ca-file | 应设置为合适值 | 是 |
| 5.6.2.5 | kubelet | --anonymous-auth | 应设置为false或 在 config file文件中配置 | 是 |
| 5.6.2.6 | kubelet | --RotateKubeletServerCertificate | 应设置为true | 否 |
| 5.6.2.7 | kubelet | --authorization-mode | 应未设置为AlwaysAllow，参数处于deprecated状态，推荐使用config file文件来设置 | 是 |

### 5.6.3 防止拒绝服务

kubelet 需能正确配置 kubelet 与其他服务通信的最大空闲时间以防止拒绝服务攻击，具体要求见下：

表 19

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.6.3.1 | kubelet | --stream-connection-idle-timeout | 不应设置为0或在config file文件中配置 | 否 |
| 5.6.3.2 | kubelet | --pids-max-pids | 宜设置容器内的进程数限制 | 否 |

### 5.6.4 SSL/TLS 配置

kubelet 发起 TLS 协商应启用强加密套件，以保障通信加密，具体要求见下：

表 20

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.6.4.1 | kubelet | --tls-cipher-suites | 应使用强加密套件 | 否 |

### 5.6.5 系统安全

内核参数通常由系统管理员在将系统投入生产之前调整和加固，通过这些参数保护内核和系统，kubelet 启动时应仅能修改安全的内核参数，具体要求见下：

表 21

| 序号 | 对象 | 对象包含的参数 | 对参数的要求 | 自动化 |
|---|---|---|---|---|
| 5.6.5.1 | kubelet | --protect-kernel-defaults | 应设置为true，如启动参数未配置，请在--config所指的配置文件配置 | 是 |
| 5.6.5.2 | kubelet | --make-iptables-util-chains | 应设置为true，如启动参数未配置，请在--config所指的配置文件配置 | 是 |
| 5.6.5.3 | kubelet | --hostname-override | 宜未设置 | 否 |
| 5.6.5.4 | kubelet | --event-qps | 宜设置为0或是以保障捕获适当事件的级别，如启动参数未配置，请在--config所指的配置文件配置 | 否 |
| 5.6.5.5 | kubelet | --allowed-unsafe-sysctls | 应禁止设置，如启动参数未配置，请在--config所指的配置文件配置 | 否 |

## 5.7 工作负载安全配置要求

### 5.7.1 镜像安全

**● 镜像创建及部署安全**

Dockerfile 是一个包含多个自定义命令的文本文件，用于构建自定义镜像。Dockerfile 的正确配置则可以保证镜像创建的安全。在镜像部署时，具备一定的准入策略，可以保证镜像部署时的安全。具体要求见下：

表 22

| 序号 | 对象 | 要求 | 自动化 |
|---|---|---|---|
| 5.7.1.1 | Image | 宜具备合适的签名检验机制 | 否 |
| 5.7.1.2 | Image | 镜像发布到生产前应是经过漏洞扫描且修复漏洞的，重建镜像时及时更新安全补丁 | 否 |
| 5.7.1.3 | Image | 宜使用ImagePolicyWebhook准入控制插件配置镜像溯源(Image Provenance) | 否 |
| 5.7.1.4 | Image | 宜使用最小的经过安全加固的基础镜像，避免安装或运行不必要的软件或命令，以减少攻击面。 | 否 |
| 5.7.1.5 | Dockerfile | 应确保部署镜像的版本（tag）不为latest | 是 |

表 22（续）

| 序号 | 对象 | 要求 | 自动化 |
|---|---|---|---|
| 5.7.1.6 | Dockerfile | 在创建镜像时，应确保镜像只暴露必要的服务端口 | 否 |
| 5.7.1.7 | Dockerfile | 不应存储敏感信息，比如秘钥等 | 否 |
| 5.7.1.8 | Dockerfile | 宜使用官方的、可信的镜像作为基础镜像 | 否 |
| 5.7.1.9 | Dockerfile | 宜在镜像中移除setuid和setgid权限 | 否 |
| 5.7.1.10 | Dockerfile | 应优先选择COPY而不是ADD | 否 |
| 5.7.1.11 | Dockerfile | 应确保镜像构建文件中不存在弃用命令，镜像构建文件中有些构建命令已弃用，如：MAINTAINER | 是 |
| 5.7.1.12 | Dockerfile | 避免单独或者在单行命令中使用update更新指令 | 是 |

- **镜像仓库安全**

镜像仓库是用来存储镜像文件的，确保镜像仓库的安全则可以避免对镜像的变相攻击。具体要求见下：

表 23

| 序号 | 对象 | 要求 | 自动化 |
|---|---|---|---|
| 5.7.1.13 | 镜像仓库 | 应确保镜像仓库配置自动化的安全扫描 | 否 |
| 5.7.1.14 | 镜像仓库 | 应支持周期性扫描库内镜像 | 否 |
| 5.7.1.15 | 镜像仓库 | 应确保最小化的镜像仓库访问权限，尤其限制推送权限的授权范围 | 否 |
| 5.7.1.16 | 镜像仓库 | 宜通过策略治理等方式限制在集群中可拉取的可信镜像仓库白名单 | 否 |
| 5.7.1.17 | 镜像仓库 | 应确保具备镜像仓库自身风险扫描的能力，如漏洞，弱口令等 | 否 |
| 5.7.1.18 | 镜像仓库 | 应确保对镜像仓库进行细粒度的访问控制，如IP、角色、项目空间等 | 否 |
| 5.7.1.19 | 镜像仓库 | 镜像仓库应具备镜像文件加密传输的能力 | 是 |

### 5.7.2 启动安全

Kubernetes 的 Yaml 文件中通常存在对 Pod 所使用的网络、CPU、内存及宿主机资源的参数，若此类参数能够被正确配置，则可以减少 Pod 在运行时的风险，具体要求见下：

表 24

| 序号 | 对象 | 要求 | 自动化 |
|---|---|---|---|
| 5.7.2.1 | Pod | 宜启用强制访问控制 | 否 |
| 5.7.2.2 | Pod | 应确保最小化root用户容器准入 | 否 |
| 5.7.2.3 | Pod | 宜确保最小化添加了capabilities的容器的准入 | 否 |
| 5.7.2.4 | Pod | 应最小化使用特权容器 | 否 |
| 5.7.2.5 | Pod | 应禁止容器挂载主机的敏感目录到容器中，特别是在读写模式下 | 否 |
| 5.7.2.6 | Pod | 应确保容器禁止主机上1024以下的特权端口映射到容器内 | 否 |
| 5.7.2.7 | Pod | 应确保容器只开启必要的端口 | 否 |
| 5.7.2.8 | Pod | 应确保容器的根文件系统以只读方式挂载 | 否 |
| 5.7.2.9 | Pod | 应最小化容器共享主机的网络命名空间的准入 | 是 |

表 24（续）

| 序号 | 对象 | 要求 | 自动化 |
|---|---|---|---|
| 5.7.2.10 | Pod | 应最小化容器共享主机的IPC命令空间的准入，即不宜设置hostIPC | 是 |
| 5.7.2.11 | Pod | 应最小化容器共享主机的进程ID命令空间的准入，即不宜设置hostPID | 是 |
| 5.7.2.12 | Pod | 宜禁止容器共享主机设备 | 是 |
| 5.7.2.13 | Pod | 应最小化使用NET_RAW功能的容器准入 | 是 |
| 5.7.2.14 | Pod | 应确保启动容器时不挂载运行时套接字文件 | 否 |
| 5.7.2.15 | Pod | 宜在Pod规格文件中应当将seccomp配置文件设置为runtime/default | 否 |
| 5.7.2.16 | Pod | 应确保最小化允许特权提升的容器准入 | 是 |
| 5.7.2.17 | Pod | 应确保最小化Windows主机进程(HostProcess)容器准入 | 否 |
| 5.7.2.18 | Pod | 应确保最小化HostPath卷的准入 | 否 |
| 5.7.2.19 | Pod | 应确保最小化使用HostPorts的容器的准入 | 否 |

### 5.7.3 身份认证和访问控制

Service Account 是一种机密帐户，在 Kubernete 集群中为工作负载提供身份。为确保工作负载的安全，所有需要访问 Kubernetes API 的工作负载都需要创建显式服务帐户并基于 RBAC 最小化分配权限。

表 25

| 序号 | 对象 | 要求 | 自动化 |
|---|---|---|---|
| 5.7.3.1 | Service Account | 应确保未主动使用默认服务帐户(service accounts) | 否 |
| 5.7.3.2 | Service Account | 应确保只在必要时挂载服务帐户令牌(token) | 否 |
| 5.7.3.3 | Service Account | 宜最小化 Service Account 的权限 | 否 |
| 5.7.3.4 | ClusterRoles/ Roles | 应确保避免使用 system:masters 组 | 否 |
| 5.7.3.5 | ClusterRoles/ Roles | 应限制 Kubernetes 集群中 Bind、Impersonate 和 Escalate 权限的使用 | 否 |
| 5.7.3.6 | ClusterRoles/ Roles | 应确保仅在需要时使用 cluster-admin 角色 | 否 |
| 5.7.3.7 | ClusterRoles/ Roles | 应尽量减少 Roles 和 ClusterRoles 中通配符的使用 | 否 |

### 5.7.4 防止信息泄露

Kubernetes 提供了 Secret 资源来保存敏感信息，Secret 中的 data 信息应加密存在，且最小化 secrets 的访问权限。若有更复杂的秘密管理需求，可以考虑使用外部 secrets storage 和管理系统，而不是直接使用 Kubernetes Secrets。具体要求见下：

表 26

| 序号 | 对象 | 要求 | 自动化 |
|---|---|---|---|
| 5.7.4.1 | secret | 宜避免使用环境变量方式存储密文等敏感信息（secret） | 否 |
| 5.7.4.2 | secret | 宜使用外部机密管理系统保存secret资源 | 否 |
| 5.7.4.3 | secret | 应确保密文等敏感信息加密后以secret资源类型进行存储 | 否 |
| 5.7.4.4 | secret | 应确保Kubernetes集群中证书私钥加密存储 | 否 |
| 5.7.4.5 | secret | 应对secret访问最小化 | 否 |

### 5.7.5 资源配额

Kubernetes 支持在工作负载和 Namespace 级别设置资源配额，合理设置资源配额有助于防止 DDos 攻击。具体要求见下：

表 27

| 序号 | 对象 | 要求 | 自动化 |
|---|---|---|---|
| 5.7.5.1 | Pod | 宜设置CPU资源请求(resources.requests.cpu)及资源限制(resources.limits.cpu)，并根据pod运行优先级为pod中每个容器正确设置资源请求和资源限制的匹配关系 | 否 |
| 5.7.5.2 | Pod | 宜设置内存请求(resources.requests.memory)及内存限制(resources.limits.memory)，并根据pod运行优先级为pod中每个容器正确设置资源请求和资源限制的匹配关系 | 否 |
| 5.7.5.3 | Namespace | 宜限制每个命名空间能够分配的资源总量，控制的资源包括： CPU、内存、存储、Pods、Services、Deployments、Statefulsets、Daemonsets、PV、PVC、Volume、Storageclass等 | 否 |
| 5.7.5.4 | Namespace | 宜配置命名空间下容器的Limit ranges | 是 |
| 5.7.5.5 | Pod | 宜谨慎在Service中使用NodePort，避免绕过NetworkPolicy，若必须使用，宜定义单独的NetworkPolicy，并结合使用LoadBalancer或Ingress来提高可用性和扩展性 | 否 |

### 5.7.6 其他

合理使用 Namespace 来隔离 Kubernetes 对象，限制用户权限的范围可以减少错误或恶意活动的影响。具体要求见下：

表 28

| 序号 | 对象 | 要求 | 自动化 |
|---|---|---|---|
| 5.7.6.1 | Namespace | 应使用命名空间来创建资源之间的管理边界 | 否 |
| 5.7.6.2 | Namespace | 宜避免使用默认命名空间 | 否 |
| 5.7.6.3 | Policy | 应确保集群至少有一个有效的策略控制机制 | 否 |

## 5.8 CNI 插件和网络策略安全配置要求

### 5.8.1 文件目录安全

Kubernetes 网络策略一般由 CNI 插件来执行，因此应控制 CNI 插件和配置文件的属主和权限，确保只有管理员，或其他具备系统管理员权限的用户才能修改该文件。具体要求见下：

表 29

| 序号 | 对象 | 要求 | 自动化 |
| --- | --- | --- | --- |
| 5.8.1.1 | CNI插件 | 应确保容器网络接口文件的权限设置为600或更严格； | 是 |
| 5.8.1.2 | CNI插件 | 应确保容器网络接口文件所属设置为root:root； | 是 |

### 5.8.2 网络策略安全配置

使用网络策略隔离集群网络中的流量，在同一 Kubernete 集群上运行不同的应用程序会导致一个受损应用程序攻击相邻应用程序的风险。具体要求见下：

表 30

| 要求序号 | 对象 | 要求 | 自动化 |
| --- | --- | --- | --- |
| 5.8.2.1 | NetworkPolicy | 宜确保使用的CNI插件支持网络策略(Network Policies) | 否 |
| 5.8.2.2 | NetworkPolicy | 宜设置合理的NetworkPolicy资源，以确保pod之间通信安全，还应确保所有 NetworkPolicy 至少针对一个 Pod | 否 |
| 5.8.2.3 | NetworkPolicy | 宜确保所有的命名空间都定义了网络策略 | 否 |

# 附 录 A
# （资料性附录）
# 检查方法和修复方法

附录 A 为第 5 章所述要求的检查方法和修复方法，其中表格序号列对应第 5 章所述要求，例如序号 5.1.1.1 对应第 5 章要求 5.1.1.1。本文件所述检查方法和修复方法仅供参考。

## A.1 API Server 安全配置

### A.1.1 文件目录安全

表 A. 1

| 序号 | 检查方法 | 修复方法 |
|---|---|---|
| 5.1.1.1 | 在k8s master节点运行如下命令进行检查，验证文件权限是否为600或者更严格<br>stat -c %a /etc/kubernetes/admin.conf | 参考如下命令，在master节点上执行命令将文件权限修改为600。<br>chmod 600 /etc/kubernetes/admin.conf |
| 5.1.1.2 | pod描述文件在k8s master节点运行如下命令进行检查，验证文件属主是否为root:root<br>1stat -c %U:%G /etc/kubernetes/manifests/kube-apiserver.yaml | 参考如下命令，在 master 节点上执行命令修改文件属主为 root:root，默认情况下 kube-apiserver.yaml 文件属主为 root:root。<br>chown root:root /etc/kubernetes/manifests/kube-apiserver.yaml |
| 5.1.1.3 | 在 k8s master 节点运行如下命令进行检查，验证文件权限是否为 600 或者更严格。<br>ls -laR /etc/kubernetes/pki/*.crt | 可参考如下命令，在master节点上执行命令将文件权限修改为600，默认情况下*.crt文件的权限为644。<br>chmod 600 /etc/kubernetes/pki/*.crt |
| 5.1.1.4 | 在 k8s master 节点运行如下命令进行检查，验证文件属主是否为 root:root<br>ls -laR /etc/kubernetes/pki/ | 可参考如下命令，在 master 节点上执行命令修改文件属主为 root:root，默认情况下/etc/kubernetes/pki/目录以及其中包含的所有文件属主为 root:root。<br>chown -R root:root /etc/kubernetes/pki/ |

### A.1.2 身份认证和访问控制

表 A. 2

| 序号 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.1.2.1 | Master 节点上执行如下命令，确认 kube-apiserver 的启动参数“--anonymous-auth”已经被配置为“false”<br>ps -ef \|grep kube-apiserver | 修改 Master 节点上 kube-apiserver 的 pod 描述文件/etc/kubernetes/manifests/kube-apiserver.yaml，设置如下参数：<br>--anonymous-auth=false |
| 5.1.2.2 | Master 节点上执行如下命令，确认不存在“--token-auth-file”参数<br>ps -ef \|grep kube-apiserver | 参照官方文档使用其它可替代的认证方式，然后去除 Master 节点上 apiserver 的描述文件“/etc/kubernetes/manifests/kube-apiserver.yaml”中的--token-auth-file=<filename>参数。<br>默认未配置“--token-auth-file”参数 |
| 5.1.2.3 | 在 Master 节点上执行以下命令，验证--authorization-mode 参数存在并且未设置为 AlwaysAllow<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kube-apiserver.yaml，并将“--authorization-mode”参数设置为 AlwaysAllow 以外的值，如，RBAC，Node，ABAC，Webhook，具体举例见下：<br>--authorization-mode = RBAC |
| 5.1.2.4 | 在 Master 节点上执行以下命令，检查“--authorization-mode”参数是否存在并且值中包括“Node”<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kube-apiserver.yaml，并将--authorization-mode 参数设置为包含 Node 的值，如：<br>--authorization-mode=Node,RBAC |
| 5.1.2.5 | 在Master节点上执行以下命令，检查“--authorization-mode”参数是否存在并且值中包括“RBAC”<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kube-apiserver.yaml，并将--authorization-mode 参数设置为包含 RBAC 的值，例如：<br>--authorization-mode=Node,RBAC |
| 5.1.2.6 | 在 Master 节点上执行以下命令，检查“--service-account-lookup”是否设置为“true”<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kube-apiserver.yaml，并设置以下参数。或者可以从此文件中删除--service-account-lookup 参数，以使默认设置生效。<br>--service-account-lookup=true |
| 5.1.2.7 | 在Master节点上执行以下命令，检查“--service-account-key-file”是否指定文件 | 在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kube-apiserver.yaml，并将--service-account-key-file 参数设置为服务帐户的公钥文件--service-account-key-file=<key-file-filename> |

表 A.2（续）

| 序号 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.1.2.8 | 在Master节点上执行以下命令，确保已正确配置<br>--kubelet-client-certificate 和<br>--kubelet-client-key 参数<br>ps -ef \|grep kube-apiserver | 在Master节点上的/etc/kubernetes/manifests/kube-apiserver.yaml并设置kubelet客户端证书和关键参数见下：<br>--kubelet-client-certificate=<path/to/client-certificate-file><br>--kubelet-client-key=<path/to/client-key-file> |
| 5.1.2.9 | 在Master节点上执行以下命令，确保已正确配置“--kubelet-certificate-authority”<br>ps -ef \|grep kube-apiserver | 编辑apiserver的pod规范文件，并将--kubelet-certificate-authority参数设置为ca文件路径。<br>--kubelet-certificate-authority=<ca-string> |
| 5.1.2.10 | 在Master节点上执行以下命令，检查“--tls-cert-file”和“--tls-private-key-file”是否已正确配置证书和私钥<br>ps -ef \|grep kube-apiserver | 在Master节点上编辑apiserver的pod规范文件/etc/kubernetes/manifests/kube-apiserver.yaml，并设置TLS证书和私钥文件参数。<br>--tls-cert-file=<path/to/tls-certificate-file><br>--tls-private-key-file=<path/to/tls-key-file> |
| 5.1.2.11 | 在Master节点上执行以下命令，检查“client-ca-file”参数是否已正确配置。<br>ps -ef \|grep kube-apiserver | 在Master节点上编辑apiserver的pod规范文件/etc/kubernetes/manifests/kube-apiserver.yaml并设置客户端ca文件。<br>--client-ca-file=<path/to/client-ca-file> |
| 5.1.2.12 | 在Master节点上执行以下命令，检查“--etcd-certfile”和“--etcd-keyfile”是否已正确配置证书和私钥<br>ps -ef \|grep kube-apiserver | 在Master节点上编辑apiserver的pod规范文件/etc/kubernetes/manifests/kube-apiserver.yaml并设置etcd证书和密钥文件参数。<br>--etcd-certfile=<path/to/client-certificate-file><br>--etcd-keyfile=<path/to/client-key-file> |
| 5.1.2.13 | 在Master节点上执行以下命令，检查“etcd-cafile”是否已正确配置<br>ps -ef \|grep kube-apiserver | 在Master节点上编辑apiserver的的pod规范文件/etc/kubernetes/manifests/kube-apiserver.yaml并设置etcd证书ca文件参数。<br>--etcd-cafile=<path/to/ca-file> |
| 5.1.2.14 | 在Master节点上执行以下命令，检查“--enable-admission-plugins”参数是否存在并且值中包括“EventRateLimit”<br>ps -ef \|grep kube-apiserver | 编辑apiserver的pod规范文件/etc/kubernetes/manifests/kube-apiserver.yaml并设置以下参数：<br>--enable-admission-plugins=...,EventRateLimit,...<br>--admission-control-config-file=<path/to/configuration/file> |
| 5.1.2.15 | 在Master节点上执行以下命令，查看是否设置了 | 在Master节点上编辑apiserver的pod规范文件/etc/kubernetes/manifests/kube-apiserver.yaml，然后删 |

| | | |
|---|---|---|
| | --enable-admission-plugins参数，若设置了，则其值不应包括AlwaysAdmit<br>ps -ef \|grep kube-apiserver | 除--enable-admission-plugins参数，或者将其设置为不包含AlwaysAdmit的值 |

表 A.2（续）

| 序号 | 检查方法 | 修复方法 |
|---|---|---|
| 5.1.2.16 | 在Master节点上执行以下命令，检查“--enable-admission-plugins”参数被设置并且其值中包含了“AlwaysPullImages”<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件 /etc/kubernetes/manifests/kube-apiserver.yaml，并将 --enable-admission-plugins 参数设置为包括 AlwaysPullImages |
| 5.1.2.17 | 在Master节点上执行以下命令，检查“--enable-admission-plugins”参数被设置并且其值中包含了“SecurityContextDeny”<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件 /etc/kubernetes/manifests/kube-apiserver.yaml，若不使用 PodSecurityPolicy，将--enable-admission-plugins 参数需要开启 SecurityContextDeny，如<br>--enable-admission-plugins=...,SecurityContextDeny,... |
| 5.1.2.18 | 在Master节点上执行以下命令，检查“--disable-admission-plugins”参数被设置并且其值中不包含“ServiceAccount”<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件 /etc/kubernetes/manifests/kube-apiserver.yaml，并确保 --disable-admission-plugins 参数设置为不包含 ServiceAccount的值 |
| 5.1.2.19 | 在 Master 节点上执行如下命令，检查“--disable-admission-plugins”参数被设置并且其值中不包含 NamespaceLifecycle<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件 /etc/kubernetes/manifests/kube-apiserver.yaml，并确保 --disable-admission-plugins 参数不包含 NamespaceLifecycle |
| 5.1.2.20 | 在Master节点上执行以下命令，检查“--enable-admission-plugins”参数被设置并且其值中包含了“NodeRestriction”<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件 /etc/kubernetes/manifests/kube-apiserver.yaml，并将 --enable-admission-plugins 参数设置为包含 NodeRestriction 的值：<br>--enable-admission-plugins=...,NodeRestriction,... |
| 5.1.2.21 | 在 Master 节点上执行如下命令，检查“--DenyServiceExternalIPs”参数应未设置<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件 /etc/kubernetes/manifests/kube-apiserver.yaml，删除 --DenyServiceExternalIPs参数；或在 kube-apiserver 的 disable-admission-plugins 参数中包含了 DenyServiceExternalIPs |

A.1.3 防止拒绝服务攻击

表 A. 3

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.1.3.1 | 在Master节点上执行ps -ef \|grep kube-apiserver 命令，检查“request-timeout”是否配置以及是否是一个合适的值 | 在 Master 节点上编辑 apiserver 的 pod 规范文件 /etc/kubernetes/manifests/kubeapiserver.yaml，并根据需要设置以下参数，推荐 60s。 例如，<br>--request-timeout=60s |

A.1.4 防止信息泄露

表 A. 4

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.1.4.1 | Master 节点上执行以下命令，检查“profiling”是否配置为“false”<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kubeapiserver.yaml，并设置以下参数：<br>--profiling=false |
| 5.1.4.2 | 在 Master 节点上执行以下命令，检查“--encryption-provider-config”“--experimental-encryption-provider-config “是否已正确配置，确保已涵盖 secret 等期望被加密存储的资源<br>ps -ef \|grep kube-apiserver | 遵循 Kubernetes 文档并配置 EncryptionConfig 文件。然后，在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kube-apiserver.yaml，并将--encryption-provider-config 参数设置为该文件的路径：<br>--encryption-provider-config=</path/to/EncryptionConfig/File> |
| 5.1.4.3 | 在 Master 节点上执行以下命令，验证--secure-port 参数是否未设置或设置为介于 1 和 65535 之间的整数值<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kube-apiserver.yaml，将--secure-port 参数设置为其他所需的安全端口 |
| 5.1.4.4 | 在 Master 节点上执行以下命令，确认--insecure-bind-address 参数不存在<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kube-apiserver.yaml，并删除--insecure-bind-address 参数 |
| 5.1.4.5 | 在 Master 节点上执行以下命令，检查“--insecure-port”是否配置为“0”<br>ps -ef \|grep kube-apiserver | 在在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kube-apiserver.yaml 并设置以下参数，<br>--insecure-port = 0 |

A.1.5 日志安全

表 A. 5

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.1.5.1 | 在 Master 节点上执行以下命令，检查“audit-log-path”是否配置了正确的路径<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件 /etc/kubernetes/manifests/kube-apiserver.yaml，并将--audit-log-path 参数设置为要写入审核日志的适当路径和文件，例如：<br>--audit-log-path=/var/log/apiserver/audit.log |
| 5.1.5.2 | 在 Master 节点上执行以下命令，检查“audit-log-maxage”参数是否设置为 30 或适当的设置<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件 /etc/kubernetes/manifests/kube-apiserver.yaml，并将--audit-log-maxage 参数设置为 30 或适当的天数：<br>--audit-log-maxage=30 |
| 5.1.5.3 | 在 Master 节点上执行以下命令，检查“audit-log-maxbackup”参数是否设置为 10 或适当的设置<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件 /etc/kubernetes/manifests/kube-apiserver.yaml，并将--audit-log-maxbackup 参数设置为 10 或适当的值。<br>--audit-log-maxbackup=10 |
| 5.1.5.4 | 在 Master 节点上执行以下命令，检查“audit-log-maxsize”参数是否设置为 100 或适当的设置<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kube-apiserver.yaml 并将--audit-log-maxsize 参数设置为适当的大小，以 MB 为单位。例如，将其设置为 100 MB：<br>--audit-log-maxsize=100 |
| 5.1.5.5 | 在其中一个集群主节点上执行以下命令：ps -ef \| grep kube-apiserver，检查是否设置了--audit-policy-file。检查指定文件的内容，并确保其中包含有效的审核策略<br>ps -ef \|grep kube-apiserver | 为集群创建审计策略文件。<br>默认集群中未创建审计策略文件，除非指定--audit-policy-file 标志，否则不会执行审计。 |

A.1.6 SSL/TLS 配置

表 A. 6

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.1.6.1 | 在 Master 节点上执行以下命令，确保“--kubelet-https”参数值为“true”或者参数不存在<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kube-apiserver.yaml 并删除--kubelet-https 参数 |
| 5.1.6.2 | 在 Master 节点上执行以下命令，确保“--tls-cipher-suites”参数值是否为 HIGH 的加密套件。<br>ps -ef \|grep kube-apiserver | 在 Master 节点上编辑 apiserver 的 pod 规范文件/etc/kubernetes/manifests/kube-apiserver.yaml，推荐使用 TLS1.2 或 TLS1.3 协议建立加密通道，TLS1.2 加密套件应选择安全程度为 HIGH 的加密套件，TLS1.3 的加密套件请参考 RFC 8446。 |
| 5.1.6.3 | 检查用户对集群的访问权限，并确保用户没有使用 Kubernetes 客户端证书身份验证。默认情况下，客户端证书身份验证处于启用状态。 | 使用 Kubernetes 提供的替代机制，如使用 OIDC（OpenID Connect 协议）取代客户端证书。 |

A.2 控制管理器安全配置

A.2.1 文件目录安全

表 A. 7

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.2.1.1 | 在 k8s master 节点运行如下命令进行检查，验证文件权限是否为 600 或者更严格。<br>stat -c %a /etc/kubernetes/manifests/kube-controller-manager.yaml | 如果不符合要求则对文件权限进行修改，可参考如下命令，在 master 节点上执行命令将文件权限修改为 600，默认情况下 kube-controller-manager.yaml 文件的权限为 640。<br>chmod 600 /etc/kubernetes/manifests/kube-controller-manager.yaml |
| 5.2.1.2 | 在k8s master节点运行如下命令进行检查，验证文件属主是否为root:root。<br>stat -c %U:%G /etc/kubernetes/manifests/kube-controller-manager.yaml | 如果不符合规则要求则修改文件属主，可参考如下命令，在master节点上执行命令修改文件属主为root:root，默认情况下kube-controller-manager.yaml文件属主为root:root。<br>chown root:root /etc/kubernetes/manifests/kube-controller-manager.yaml |
| 5.2.1.3 | 在k8s master节点运行如下命令进行检查，验证文件权限是否为600或者更严格。<br>stat -c %a /etc/kubernetes/controller-manager.conf<br>执行命令返回的权限结果确保为 600 或更小权限。 | 如果不符合要求则对文件权限进行修改，可参考如下命令，在master节点上执行命令将文件权限修改为600，默认情况下controller-manager.conf文件的权限为640。<br>chmod 600 /etc/kubernetes/controller-manager.conf |

表A. 7(续)

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.2.1.4 | 在k8s master节点运行如下命令进行检查，验证文件属主是否为root:root。<br>stat -c %U:%G /etc/kubernetes/controller-manager.conf<br>执行命令返回的属主结果确保 root:root。 | 如果不符合规则要求则修改文件属主，可参考如下命令，在master节点上执行命令修改文件属主为root:root，默认情况下controller-manager.conf文件属主为root:root。<br>chown root:root /etc/kubernetes/controller-manager.conf |

A.2.2 身份认证和访问控制

表 A. 8

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.2.2.1 | 在Master节点上执行如下命令，确认“--use-service-account-credentials”是否为“true”。<br>.ps -ef \|grep kube-controller-manager | 在master节点上编辑控制管理器的pod规格文件/etc/kubernetes/manifests/kube-controller-manager.yaml，配置如下参数：<br>--use-service-account-credentials=true |
| 5.2.2.2 | 在Master节点上执行如下命令，确认“--service-account-private-key-file”是否正确配置。<br>ps -ef \|grep kube-controller-manager | 在master节点上编辑控制管理器的pod规格文件/etc/kubernetes/manifests/kube-controller-manager.yaml，将参数--service-account-private-key-file设置为业务帐号的私钥文件：<br>--service-account-private-key-file=<filename><br>“--service-account-private-key-file”默认值未设置 |
| 5.2.2.3 | 在Master节点上执行如下命令，检查“--root-ca-file”是否存在，并且该参数配置为包含apiserver服务证书根证书的证书捆绑文件。<br>ps -ef \|grep kube-controller-manager | 在master节点上编辑控制管理器的pod规格文件/etc/kubernetes/manifests/kube-controller-manager.yaml，将参数--root-ca-file设置为根ca证书文件：<br>--root-ca-file=<filename><br>“--root-ca-file”默认值未设置 |
| 5.2.2.4 | 在Master节点上执行如下命令，检查RotateKubeletServerCertificate参数是否存在，如果存在，需要确保未被设置为false。<br>ps -ef \|grep kube-controller-manager | 在master节点上编辑控制管理器的pod规格文件/etc/kubernetes/manifests/kube-controller-manager.yaml，将参数--feature-gates的值修改为包含RotateKubeletServerCertificate=true，如下所示：<br>--feature-gates=RotateKubeletServerCertificate=true<br>RotateKubeletServerCertificate默认值为true |

A.2.3 防止拒绝服务

表 A. 9

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.2.3.1 | 在Master节点上执行如下命令，确认--terminated-pod-gc-threshold是否正确配置。<br>ps -ef \|grep kube-controller-manager | 在master节点上编辑控制管理器的pod规格文件/etc/kubernetes/manifests/kube-controller-manager.yaml，将--terminated-pod-gc-threshold设置为合适的阈值，例如：<br>--terminated-pod-gc-threshold=10<br>“terminated-pod-gc-threshold”默认值为“12500” |

A.2.4 防止信息泄露

表 A. 10

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.2.4.1 | 在Master节点上执行如下命令，检查--profiling参数是否设置为false。<br>ps -ef \|grep kube-controller-manager | 在master节点上编辑控制管理器的pod规格文件/etc/kubernetes/manifests/kube-controller-manager.yaml，配置如下参数：<br>--profiling=false<br>“profiling”默认值为“true”。 |
| 5.2.4.2 | 在Master节点上执行如下命令，确认“--bind-address”参数为“127.0.0.1”。<br>1.ps -ef \|grep kube-controller-manager | 编辑控制管理器的Pod规格文件，即/etc/kubernetes/manifests/kube-controller-manager.yaml文件，确保参数--bind-address填写正确，为127.0.0.1。<br>“--bind-address”默认值为“0.0.0.0”。 |

A.3 调度器安全配置

A.3.1 文件目录安全

表 A. 11

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.3.1.1 | 在k8s master节点运行如下命令进行检查，验证文件权限是否为600或者更严格。<br>stat -c %a /etc/kubernetes/manifests/kube-scheduler.yaml | 如果不符合要求则对文件权限进行修改，可参考如下命令，在master节点上执行命令将文件权限修改为600，默认情况下ube-scheduler.yaml文件的权限为640。<br>chmod 600 /etc/kubernetes/manifests/kube-scheduler.yaml |
| 5.3.1.2 | 在k8s master节点运行如下命令进行检查，验证文件属主是否为root:root<br>stat -c %U:%G /etc/kubernetes/manifests/kube-scheduler.yaml | 如果不符合规则要求则修改文件属主，可参考如下命令，在master节点上执行命令修改文件属主为root:root，默认情况下kube-scheduler.yaml文件属主为root:root。<br>chown root:root /etc/kubernetes/manifests/kube-scheduler.yaml |

表 A. 11（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.3.1.3 | 在k8s master节点运行如下命令进行检查，验证文件权限是否为600或者更严格。<br>stat -c %a /etc/kubernetes/scheduler.conf | 如果不符合要求则对文件权限进行修改，可参考如下命令，在master节点上执行命令将文件权限修改为600，默认情况下scheduler.conf文件的权限为640。<br>stat -c %a /etc/kubernetes/scheduler.conf |
| 5.3.1.4 | 在k8s master节点运行如下命令进行检查，验证文件属主是否为root:root。<br>stat -c %U:%G /etc/kubernetes/scheduler.conf | 如果不符合规则要求则修改文件属主，可参考如下命令，在master节点上执行命令修改文件属主为root:root，默认情况下scheduler.conf文件属主为root:root。<br>chown scheduler:scheduler/etc/kubernetes/scheduler.conf |

A.3.2 防止信息泄露

表 A. 12

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.3.2.1 | 在Master节点上执行如下命令，检查--profiling参数是否设置为false。<br>ps -ef \|grep kube-scheduler | 在master节点上编辑Scheduler pod规格文件/etc/kubernetes/manifests/kube-scheduler.yaml，配置如下参数。<br>--profiling=false<br>“profiling”默认值为“true”。 |
| 5.3.2.2 | 在Master节点上执行如下命令，确认“--bind-address”参数为“127.0.0.1”。<br>ps -ef \|grep kube-scheduler | 编辑Scheduler的Pod规格文件，即/etc/kubernetes/manifests/kube-scheduler.yaml文件，确保参数--bind-address填写正确，为“127.0.0.1”。<br>“--bind-address”默认值为“0.0.0.0”。 |

A.4 etcd 安全配置

A.4.1 文件目录安全

表 A. 13

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.4.1.1 | 在 k8s master 节点运行如下命令进行检查，验证文件权限是否为 600 或者更严格<br>stat -c %a /etc/kubernetes/manifests/etcd.yaml | 如果不符合要求则对文件权限进行修改，可参考如下命令，在 master 节点上执行命令将文件权限修改为 600，默认情况下 etcd.yaml 文件的权限为 640。chmod 600 /etc/kubernetes/manifests/etcd.yaml |

表 A.13（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.4.1.2 | 在 k8s master 节点运行如下命令进行检查，验证文件属主是否为 root:root<br>stat -c %U:%G /etc/kubernetes/manifests/etcd.yaml | 如果不符合规则要求则修改文件属主，可参考如下命令，在 master 节点上执行命令修改文件属主为 root:root，默认情况下 etcd.yaml 文件属主为 root:root。<br>chown root:root /etc/kubernetes/manifests/etcd.yaml |
| 5.4.1.3 | etcd 的数据目录路径视具体情况而定，默认路径在/var/lib/etcd，也可以在 etcd 服务器节点上通过命令查看--data-dir 配置的路径。<br>ps -ef \| grep etcd<br>stat -c %a <path/to/etcd> | 在 etcd 服务器节点上执行命令将文件权限修改为 700，默认情况下 etcd 的数据目录权限为 755。<br>chmod 700 <path/to/etcd> |
| 5.4.1.4 | 文件路径视具体情况而定，默认文件路径在/var/lib/etcd，也可以在 etcd 服务器节点上通过命令查看--data-dir 配置的路径。<br>ps -ef \| grep etcd<br>stat -c %U:%G <path/to/etcd> | 在 etcd 服务器节点上执行命令修改文件属主为 etcd:etcd，默认情况下 etcd 的数据目录属主为 etcd:etcd。<br>chown etcd:etcd <path/to/etcd> |

A.4.2 身份认证和访问控制

表 A. 14

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.4.2.1 | 在 etcd 服务器节点上执行 ps -ef \| grep etcd 命令,验证--cert-file 和--key-file 参数是否设置得适当 | 按照 etcd 服务文档，配置 TLS 加密。然后，在主节点上编辑 etcd pod 规格文件/etc/kubernetes/manifests/etcd.yaml，并设置以下参数。<br>--cert-file=</path/to/ca-file><br>--key-file=</path/to/key-file><br>默认情况下，未设置 TLS 加密。 |
| 5.4.2.2 | 在 etcd 服务器节点上执行 ps -ef \| grep etcd 命令，验证--client-cert-auth 参数是否设置为 true | 在主节点上编辑 etcd pod 规格文件/etc/kubernetes/manifests/etcd.yaml，并设置以下参数。<br>--client-cert-auth="true"<br>默认情况下，etcd 服务可以被未经认证的客户端查询 |
| 5.4.2.3 | 在 etcd 服务器节点上执行 ps -ef \| grep etcd 命令，验证如果--auto-tls 参数存在，则不应该设置为 true | 在主节点上编辑 etcd pod 规格文件/etc/kubernetes/manifests/etcd.yaml，删除--auto-tls 参数或将其设置为 false。<br>--auto-tls=false<br>默认情况下，--auto-tls 设置为 false |

表 A.14（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.4.2.4 | 在 etcd 服务器节点上执行 ps -ef \| grep etcd 命令，验证--peer-cert-file 和--peer-key-file 参数是否设置得适当 | 按照 etcd 服务文档，并根据需要为您的 etcd 集群配置对等 TLS 加密。然后，在主节点上编辑 etcd pod 规格文件/etc/kubernetes/manifests/etcd.yaml，并设置以下参数。<br>--peer-client-file=</path/to/peer-cert-file><br>--peer-key-file=</path/to/peer-key-file> |
| 5.4.2.5 | 在 etcd 服务器节点上执行 ps -ef \| grep etcd 命令，验证--peer-client-cert-auth 参数是否设置为 true | 在主节点上编辑 etcd pod 规格文件/etc/kubernetes/manifests/etcd.yaml，并设置以下参数。<br>--peer-client-cert-auth=true |
| 5.4.2.6 | 在 etcd 服务器节点上执行 ps -ef \| grep etcd 命令，验证如果--peer-auto-tls 参数存在，若存在设置为 true | 编辑 master 节点上的 etcd pod 规范文件/etc/kubernetes/manifests/etcd.yaml，删除--peer-auto-tls 参数或将其设置为 false。<br>--peer-auto-tls=false |
| 5.4.2.7 | 1.在主节点上执行 ps -ef \| grep etcd 命令<br>2.在主节点上执行 ps -ef \| grep apiserver 命令，检查 apiserver 的--client-ca-file 引用的文件是否与 etcd 使用的--trusted-ca-file 不同 | 遵循 etcd 文档，并为 etcd 服务创建专用 ca。然后，在主节点上编辑 etcd pod 规范文件/etc/kubernetes/manifests/etcd.yaml，并设置以下参数。<br>--trusted-ca-file=</path/to/ca-file> |

A.4.3 SSL/TLS 配置

表 A. 15

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.4.3.1 | 执行 ps -ef \| grep etcd 命令，如果存在--cipher-suites 参数，设置为仅包含安全套件中的值 | 推荐使用 TLS1.2 或 TLS1.3 协议建立加密通道，TLS1.2 加密套件应选安全程度为 HIGH 的加密套件，TLS1.3 的加密套件请参考 RFC 8446。以 TLS1.2 举例，<br>按照 etcd 服务官方文档，配置 TLS 加密。然后，在主节点上编辑 etcd pod 规格文件/etc/kubernetes/manifests/etcd.yaml，并将--cipher-suites 参数设置为安全加密套件中的值 |

A.5 kube-proxy 安全配置

A.5.1 文件目录安全

表 A. 16

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.5.1.1 | 在节点上执行如下命令 ps -ef \|grep kube-proxy<br>，找到“--kubeconfig”指定的文件路径<br>使用如下命令，确认文件权限是否为 600 或者更严格。默认情况下，文件权限为 640。<br>stat -c %a <proxy kubeconfig file> | 在每个工作节点上运行下面的命令（根据系统上的文件位置）。例如：<br>chmod 600 <proxy kubeconfig file> |
| 5.5.1.2 | 在节点上执行如下命令 ps -ef \|grep kube-proxy<br>，找到“--kubeconfig”指定的文件路径<br>使用如下命令确认文件属主是否为 root:root，默认情况下，该文件属主为 root:root.<br>stat -c %a <proxy kubeconfig file> | 在每个工作节点上运行下面的命令（根据系统上的文件位置）。例如：<br>chown root:root <proxy kubeconfig file> |

## A.6 kubelet 安全配置

### A.6.1 文件目录安全

表 A. 17

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.6.1.1 | 在节点上执行如下命令，执行命令返回的权限结果确保为 600 或更小权限。默认情况下，文件权限为 640。<br>stat -c %a /etc/systemd/system/kubelet.service.d/10-kubeadm.conf | 在每个工作节点上运行下面的命令（基于系统中的文件位置），例如：<br>chmod 600 /etc/systemd/system/kubelet.service.d/kubeadm.conf |
| 5.6.1.2 | 在节点上执行如下命令，执行命令返回文件属主为 root:root。<br>stat -c %a /etc/systemd/system/kubelet.service.d/10-kubeadm.conf | 在每个工作节点上运行下面的命令（基于系统中的文件位置），例如：<br>chown root:root /etc/systemd/system/kubelet.service.d/kubeadm.conf |
| 5.6.1.3 | 请执行以下命令查询--kubeconfig 对应的 kubelet.conf 文件的路径，默认情况下，文件位于“/etc/kubernetes/”路径下。<br>ps -ef \| grep kubelet<br>使用如下命令查看文件权限为 600 或者更严格。默认情况下，文件权限为 640。<br>stat -c %a /etc/kubernetes/kubelet.conf | 在每个工作节点上运行下面的命令（基于系统中的文件位置），例如：<br>chmod 600 /etc/kubernetes/kubelet.conf |
| 5.6.1.4 | 执行 5.6.1.3 命令查询文件路径后，使用如下命令查看文件属主是否为 root:root<br>stat -c %a /etc/kubernetes/kubelet.conf | 在每个工作节点上运行下面的命令（基于系统中的文件位置），例如：<br>chown root:root /etc/kubernetes/kubelet.conf |
| 5.6.1.5 | 执行如下命令查到“--client-ca-file”参数指定的文件路 | 默认情况下，--client-ca-file 未设置。 |

表 A.17（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.6.1.5 | 径。<br>ps -ef \|grep kubelet<br>执行如下命令，确认返回的权限不大于 600。<br>stat -c %a <filename> | 执行以下命令，修改--client-ca-file 文件权限:<br>chmod 600 <filename> |
| 5.6.1.6 | 执行如下命令查到“--client-ca-file”参数指定的文件路径。<br>ps -ef \|grep kubelet<br>执行如下命令，确认返回的属主为 root:root。<br>stat -c %a <filename> | 默认情况下，--client-ca-file 未设置。<br>执行以下命令，修改--client-ca-file 文件属主:<br>chown root:root <filename> |
| 5.6.1.7 | 在业务节点上执行如下命令查看“--config”指定的文件路径<br>ps -ef \| grep kubelet \| grep config<br>使用命令查看文件权限是否为 600 或者更严格。<br>stat -c %a <kubelet config> | 默认情况下，/var/lib/kubelet/config.yaml 由 kubeadm 创建，权限为 644。<br>执行如下命令（使用检查方法中识别的 config 文件位置）：<br>chmod 600 <kubelet config>(eg. /var/lib/kubelet/config.yaml) |
| 5.6.1.8 | 在业务节点上执行如下命令查看“--config”指定的文件路径<br>ps -ef \| grep kubelet \| grep config<br>使用命令查看文件属主是否为 root:root。<br>stat -c %a <kubelet config> | 默认情况下属主为 root:root。<br>执行如下命令（使用检查方法中识别的 config 文件位置）：<br>chown root:root <kubelet config>(eg. /var/lib/kubelet/config.yaml) |

A.6.2 身份认证和访问控制

表 A. 18

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.6.2.1 | 在每个节点上执行 ps -ef \| grep kubelet 命令，验证--read-only-port 参数是否存在，如果存在，是否正确设置为 0。<br>如果不存在--read-only-port 参数，请检查是否存在由--config 指定的 Kubelet 配置文件。检查如果文件中存在 readOnlyPort 条目，则将其设置为 0。 | 默认情况下，--read-only-port 设置为 10255<br>1. 如果使用 kubelet config 文件，编辑该文件，将“--read-only-port”设置为 0。<br>2. 如果使用可执行参数，则在每个 worker 节点上编辑 kubelet 服务文件 /etc/kubernetes/kubelet.conf，在 KUBELET_SYSTEM_PODS_ARGS 变量中设置如下参数：--read-only-port=0。<br>3. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |
| 5.6.2.2 | 在每个节点上执行 ps -ef \| grep kubelet 命令，验证--rotate-certificates 参数是否不存在，如果存在，是否正确设置为 true。<br>如果--rotate-certificates 参数不存在，请验证如果存在由--config 指定的 Kubelet 配置文件，则该文件不包含 rotateCertificates: false | 1. 如果使用 kubelet config 文件，编辑该文件，添加“rotateCertificates:true”行，或者完全移除，使用默认值。<br>2. 如果使用可执行参数，则在每个 worker 节点上编辑 kubelet 服务文件 /etc/kubernetes/kubelet.conf，在 KUBELET_CERTIFICATE_ARGS 变量中变量中设置如下参数：--rotate-certificates=true<br>3. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |
| 5.6.2.3 | 集群节点上执行命令 ps -ef \| grep kubelet<br>查看参数“--tls-cert-file”和“--tls-private-key-file”的配置。<br>如果这些参数不存在，请检查是否存在由--config 指定的 Kubelet 配置，并且它包含 tlsCertFile 和 tlsPrivateKey 的适当设置。 | 1. 如果使用 kubelet config 文件，编辑该文件，将 tlsCertFile 设置为要用于标识该 kubelet 的证书文件位置，tlsPrivateKeyFile 设置为对应的私钥文件位置。<br>2. 如果使用可执行参数，则在每个 worker 节点上编辑 kubelet 服务文件 /etc/kubernetes/kubelet.conf ，在 KUBELET_CERTIFICATE_ARGS 变量中设置如下参数：<br>--tls-cert-file=<path/to/tls-certificate-file> 、<br>--tls-private-keyfile=<path/to/tls-key-file>。<br>3. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |

表 A.18（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.6.2.4 | 如果不存在--client-ca-file 参数，请检查是否存在由--config 指定的 Kubelet 配置文件，并且该文件是否将身份验证：x509: clientCAFile 设置为 ca 文件的位置。<br>ps -ef \| grep kubelet | 1. 如果使用 kubelet config 文件，编辑该文件，将“authentication: x509: clientCAFile”设置为 ca 文件的位置。<br>2. 如果使用可执行参数，则在每个 worker 节点上编辑 kubelet 服务文件 /etc/kubernetes/kubelet.conf，在 KUBELET_AUTHZ_ARGS 变量中设置如下参数：--client-ca-file=<path/to/client-ca-file>。<br>3. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |
| 5.6.2.5 | 在每个工作节点上执行以下命令，验证 --anonymous-auth 参数是否设置为 false。<br>如果不存在--anonymous-auth 参数，请检查是否存在由--config 指定的 Kubelet 配置文件，并查看匿名参数 authentication:anonymous: enabled 值是否设置为 false。<br>ps -ef \| grep kubelet | 默认情况下，kubelet 服务器允许匿名访问。<br>1. 如果使用 kubelet config 文件，编辑该文件，将“authentication:anonymous: enabled”设置为“false”。<br>2. 如果使用可执行参数，则在每个 worker 节点上编辑 kubelet 服务文件/etc/kubernetes/kubelet.conf。<br>在 KUBELET_SYSTEM_PODS_ARGS 变量中设置如下参数：--anonymous-auth=false，注意"anonymous-auth"默认值为“true”。<br>3. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |
| 5.6.2.6 | 如果 kubelet 配置文件中的 serverTLSBootstrap 为 true，或者 kubelet config 文件上设置了 --rotate-server-certificates 参数，则忽略此检查。<br>在每个节点上执行 ps -ef \| grep kubelet 命令：验证是否存在 RotateKubeletServerCertificate 参数并将其设置为 true | 1.在每个 worker 节点上编辑 kubelet 服务文件 /etc/kubernetes/kubelet.conf，设置：--rotate-certificates=true，在 KUBELET_CERTIFICATE_ARGS 变量中配置如下参数：--feature-gates=RotateKubeletServerCertificate=true<br>2. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |

表 A.18（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.6.2.7 | 在每个节点上执行 ps -ef \| grep kubelet 命令，验证--authorization-mode 参数，禁止设置 AlwaysAllow。如果不存在--authorization-mode 参数，请检查是否存在由--config 指定的 Kubelet 配置文件。查看 authorization: mode 参数值是否设置为 Webhook。 | 1. 如果使用 kubelet config 文件，编辑该文件，将 authorization: mode 设置为 Webhook。<br>2. 如果使用可执行参数，则在每个 worker 节点上编辑 kubelet 服务文件/etc/kubernetes/kubelet.conf，在 KUBELET_AUTHZ_ARGS 变量中设置如下参数：--authorization-mode=Webhook。<br>3. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |

A.6.3 防止拒绝服务

表 A. 19

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.6.3.1 | 在每个工作节点上执行 ps -ef \| grep kubelet 命令，验证--streaming-connection-idie-timeout 参数是否未设置为 0<br>如果该参数不存在，并且存在由--config 指定的 Kubelet 配置文件，请检查该文件里没有将 StreamConnectionldleTimeout 设置为 0 | “streaming-connection-idle-timeout”默认值为 4 小时。<br>1. 如果使用 kubelet config 文件，编辑该文件，将“--streaming-connection-idle-timeout”设置为满足实际业务场景的最小值。<br>2. 如果使用可执行参数，则在每个 worker 节点上编辑 kubelet 服务文件 /etc/kubernetes/kubelet.conf，在 KUBELET_SYSTEM_PODS_ARGS 变量中设置如下参数：<br>--streaming-connection-idle-timeout=5m。<br>3. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |
| 5.6.3.2 | 集群节点上执行命令 ps -ef \| grep kubelet<br>查看参数“--pod-max-pids”，如果参数不存在，请检查是否存在由--config 指定的 Kubelet 配置，该配置文件是否包含 PodPidsLimit 的适当设置。 | 1. 如果使用 kubelet config 文件，编辑该文件，添加“PodPidsLimit:1000”行，其中 1000 可根据业务场景调整，或者使用默认值。<br>2. 如果使用可执行参数，则在每个 worker 节点上编辑 kubelet 服务文件 /etc/kubernetes/kubelet.conf，在启动参数中添加--pod-max-pids 配置。<br>3. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |

A.6.4 SSL/TLS 配置

表 A. 12

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.6.4.1 | 执行 ps -ef \| grep kubelet 命令，如果存在 --tls-cipher-Suite 参数，设置为仅包含安全套件中的值。如果不存在--tls-cipher-Suite 参数，请检查是否存在由--config 指定的 Kubelet 配置文件配置是否包含 TLScipherSuite，设置为仅包括安全套件中的值 | 推荐使用 TLS1.2 或 TLS1.3 协议建立加密通道，TLS1.2 加密套件应选择安全程度为 HIGH 的加密套件，TLS1.3 的加密套件请参考 RFC 8446 |

A.6.5 系统安全

表 A. 21

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.6.5.1 | 在每个节点上执行 ps -ef \| grep kubelet 命令，检查--protect-kernel-defaults 参数是否设置为 true。<br>如果不存在--protect-kernel-defaults 参数，请检查是否存在由--config 指定的 Kubelet 配置文件，并且该文件是否将保 protectKernelDefaults 设置为 true | "protect-kernel-defaults"默认值为“false”。<br>1. 如果使用 kubelet config 文件，编辑该文件，设置 protectKernelDefaults: true。<br>2. 如果使用可执行参数，则在每个 worker 节点上编辑 kubelet 服务文件/etc/kubernetes/kubelet.conf，在 KUBELET_SYSTEM_PODS_ARGS 变量中设置如下参数：--protect-kernel-defaults=true。<br>3. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |
| 5.6.5.2 | 在每个工作节点上执行 ps -ef \| grep kubelet 命令，验证如果--make-iptables-util-chains 参数存在，则它是否设置为 true。<br>如果--make-iptables-util-chains 参数不存在，并且存在由--config 指定的 Kubelet 配置文件，请验证该文件没有将 makelPTablesUtilChains 参数设置为 false | “make-iptables-util-chains”默认值为“true”。<br>1． 如果使用 kubelet config 文件，编辑该文件，设置 makeIPTablesUtilChains: true。<br>2. 如果使用可执行参数，则在每个 worker 节点上编辑 kubelet 服务文件/etc/kubernetes/kubelet.conf，在 KUBELET_SYSTEM_PODS_ARGS 变量中去掉 --make-iptables-util-chains 参数。<br>3. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |
| 5.6.5.3 | 集群节点上执行 ps -ef \| grep kubelet 命令查看参数“--hostname-override”的配置 | 默认情况下，系统未设置，hostname-override 参数。<br>1. 编辑每个 worker 节点上的 kubelet 服务文件 /etc/kubernetes/kubelet.conf，去掉 KUBELET_SYSTEM_PODS_ARGS 变量中的 --hostname-override 参数。<br>2. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |

表 A.21（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.6.5.4 | 执行 ps -ef \| grep kubelet 命令命令，查看 --event-qps 参数的值，并确定是否已将其设置为适当值。当参数为 0 时，表示捕获所有事件。<br>如果--event-qps 参数不存在，请检查是否存在由--config 指定的 Kubelet 配置文件，并查看此位置中的值 | “event-qps”默认值为“5”。<br>1. 如果使用 kubelet config 文件，编辑该文件，将 eventRecordQPS：设置为合适的级别。<br>2. 如果使用可执行参数，则在每个 worker 节点上编辑 kubelet 服务文件/etc/kubernetes/kubelet.conf，在 KUBELET_SYSTEM_PODS_ARGS 变量中设置如下参数：--event-qps=0。<br>3. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |
| 5.6.5.5 | 在每个节点上执行以下命令，验证 --allowed-unsafe-sysctls 参数是否存在，禁止配置-allowed-unsafe-sysctls 参数。<br>ps -ef \| grep kubelet | 默认情况下，kubelet 未开启 allowedUnsafeSysctls。<br>1. 如果使用 kubelet config 文件，编辑该文件，将 allowedUnsafeSysctls 参数值置空，或移除，使用默认值；<br>2. 如果使用可执行参数，则在每个 worker 节点上编辑 kubelet 服务文件/etc/kubernetes/kubelet.conf，在 KUBELET_SYSTEM_PODS_ARGS 变量中去除 allowed-unsafe-sysctls 参数或置空。<br>3. 重启 kubelet 服务。例如：<br>systemctl daemon-reload<br>systemctl restart kubelet.service |

## A.7 工作负载安全配置

### A.7.1 镜像安全

表 A. 22

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.1.1 | 确认是否为容器镜像开启了签名验签机制 | 实现容器镜像签名验签的方式很多，典型的开源实现包括 Kritis, cosign 等。如下以 cosign 为例：<br>签名<br>cosign sign -key cosign.key dustise/sleep:v0.9.6<br>验签：<br>cosign verify -key cosign.pub repository/image:v0.9.6 |
| 5.7.1.2 | 使用商用软件或开源软件(如 Trivy, Falco 等)扫描扫描容器镜像，查看输出结果是否有漏洞 | 升级镜像替换有漏洞的组件，生成新的镜像 |

表 A. 22（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.1.3 | 宜使用 Kubernetes admission controller 来控制镜像准入<br>kube-apiserver -h \| grep enable-admission-plugins<br>查看是否有需要的 admission controller 设置 | 使用命令 kube-apiserver --enable-admission-plugins=NamespaceLifecycle,LimitRanger…<br>进行设置 |
| 5.7.1.4 | 基础镜像中若使用不必要的软件，可能会加大容器的攻击面，同时也违背了最小和精简容器镜像的概念。因此，不要为容器安装或运行任何不必要的软件。 | 精简基础镜像。 |
| 5.7.1.5 | 应确保部署镜像的版本（tag）不为 latest<br>使用 docker images 查看镜像版本是否是 latest<br>或者使用 kubectl 查看部署的负载镜像版本是否是 latest<br>kubectl get pod <pod_name> -o yaml\| grep image: | 不使用版本 latest 产生镜像<br>不使用版本 latest 设置负载中容器镜像版本 |
| 5.7.1.6 | 使用 docker inspect image，查看 NetworkSetting 中的 Ports，查看暴露的端口是否是必须的服务端口 | 修改 Dockerfile 重新生成镜像 |
| 5.7.1.7 | 查看 Dockerfile 是否存储了敏感信息如口令、密钥 | 去掉 Dockerfile 的口令、密钥后重新生成镜像 |
| 5.7.1.8 | 查看 Dockerfile 中 From 指令，检查基础镜像是否是可信的镜像<br>宜使用官方的、可信的镜像作为基础镜像 | 修改 Dockerfile，在 From 指令中使用可信的镜像 |
| 5.7.1.9 | find /bin/ -perm /+6000 -type f –exec ls –ld {}<br>查找是否有文件设置了 setuid 或 setgid 权限 | 使用 chmod a-s 对这些文件去除 setuid、setgid 权限 |
| 5.7.1.10 | 通过以下命令查看 Dockerfile 构建文件中是否存在 ADD 命令<br>grep -iE "^\s*ADD\s+.*" Dockerfile | 根据需要将 Dockerfile 中的 ADD 指令修改为 COPY 指令 |
| 5.7.1.11 | 通过以下命令查看 Dockerfile 构建文件中是否存在弃用命令<br>grep -iE "^\s*MAINTAINER\s+.*" Dockerfile | MAINTAINER 指令已经被弃用，使用 LABEL 命令替换 MAINTAINER 来添加元数据，如 LABEL maintainer="John Doe <john.doe@example.com>" |
| 5.7.1.12 | 不同的镜像仓库开启自动化安全扫描的配置不一样，应在镜像仓库或者镜像扫描服务中确认是否为镜像开启了自动化安全扫描 | 不同的镜像仓库开启自动化安全扫描的配置不一样，请查阅用户手册为镜像开启自动扫描 |

表 A.22（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.1.13/<br>5.7.1.14 | 不同的镜像仓库配置扫描周期的方式不一样，应在镜像仓库或者镜像扫描服务中确认是否配置了扫描周期 | 不同的镜像仓库开启自动化安全扫描的配置不一样，请查阅用户手册为镜像设置合理的扫描周期 |
| 5.7.1.15 | 从以下几个方面检查镜像仓库访问权限，确保镜像仓库访问权限最小化：<br>1、镜像仓库的可见性：控制哪些用户或者组织可以查看或者搜索到镜像仓库。公开的镜像仓库可以被任何人查看和拉取，但是只有拥有者或者协作者可以推送。私有的镜像仓库只能被拥有者或者指定的用户或者组织查看、拉取和推送。内部的镜像仓库只能被同一组织的成员查看、拉取和推送。为了最小化访问权限，建议将镜像仓库设置为私有或者内部，除非有特殊的需求。<br>2、镜像仓库的访问控制：控制哪些用户或者组织可以对镜像仓库进行拉取或者推送操作。为了最小化访问权限，建议将推送权限只授予必要的用户或者组织，例如开发人员或者持续集成服务。同时，建议使用不同的账号或者凭证来区分不同的操作场景，例如开发、测试、生产等。<br>3、镜像仓库的安全扫描：控制哪些类型的镜像可以被拉取或者推送到镜像仓库，以及如何检测和修复镜像中的漏洞。为了最小化访问权限，建议启用安全扫描功能，并且设置合理的策略来限制哪些类型的镜像可以被拉取或者推送，例如官方镜像、认证发布商镜像、组织自己的镜像等 | 根据需要设置镜像仓库的可见性、镜像仓库的访问控制权限、配置合理的镜像仓库扫描策略 |
| 5.7.1.16 | 检查集群是否开启了策略治理能力：<br>1、检查 K8S 集群是否启用了准入控制功能。<br>2、查看集群是否安装了策略治理组件如 Gatekeeper 或 Kyverno<br>3、检查集群是否配置了可信镜像仓库白名单策略，仅允许白名单镜像在 k8s 集群部署。<br>4、验证可信镜像仓库白名单策略是否符合预期 | 根据具体的需求开启集群策略治理能力，并配置可信镜像仓库白名单策略，确保未知镜像无法部署到集群 |

表 A.22（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.1.17 | 通过以下方面来检查镜像仓库自身风险扫描能力：<br>1、通过镜像扫描工具对镜像仓库自身的漏洞进行扫描，检查是否存在漏洞。<br>2、通过镜像扫描工具对镜像仓库自身的弱口令进行扫描，检查是否存在弱口令。<br>3、通过镜像扫描工具对镜像仓库自身的配置进行扫描，检查是否在若配置项。<br>4、通过镜像扫描工具对镜像仓库自身的恶意文件进行扫描，检查是否存在植入的恶意软件 | 解决扫描发现的问题 |
| 5.7.1.18 | 通过以下方面来对镜像仓库进行细粒度的访问控制。<br>1、使用角色或用户组或项目空间来管理不同的访问者，根据他们的身份和职责分配合适的权限<br>2、使用 IP 白名单或黑名单来控制哪些 IP 地址可以访问仓库，以防止恶意攻击或泄露敏感信息<br>3、使用 HTTPS 协议来加密仓库的通信，以保护数据的完整性和隐私性<br>4、使用 token 或证书来验证访问者的身份，以防止伪造或盗用 | 根据具体的需求对镜像仓库进行细粒度的访问控制，配置合理的策略，如配置 IP、角色、项目空间等策略，确保权限最小化 |
| 5.7.1.19 | 通过以下命令查看镜像仓库是否具备镜像文件加密传输的能力。<br>curl -v https://<镜像仓库地址> 2>&1 \| grep -iE "ssl\|tls"<br>如果命令输出显示了有效的证书和协议信息，说明该镜像仓库具备文件机密传输的能力，否则不具备 | 根据具体的镜像仓库设置镜像仓库支持 HTTPS 以加密传输镜像文件 |

A.7.2 启动安全

表 A. 23

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.2.1 | 执行以下命令，结果不匹配(AppArmorProfile=\s*$)\|(AppArmorProfile=<no value>)<br>docker ps --quiet --all\| xargs docker inspect --format '{{ .Id }}: AppArmorProfile={{ .AppArmorProfile }}' 2>/dev/null<br><br>执行以下命令，结果不匹配(SecurityOpt=\s*$)\|(SecurityOpt=<no value>)<br>docker ps --quiet --all\| xargs docker inspect --format '{{ .Id }}: SecurityOpt={{ .HostConfig.SecurityOpt }}' 2>/dev/null | 如果 Linux 操作系统能够适用 AppArmor。遵循下列步骤：1. 确认 AppArmor 已安装。如尚未安装，安装 AppArmor。2. 为 docker 容器创建或导入一个 AppArmor 配置文件。3. 将该配置文件设置为 enforcing 模式。4. 使用自定义的 AppArmor 配置文件启动 docker 容器。例如：docker run --interactive --tty --security-opt="apparmor:PROFILENAME" centos /bin/bash 或者，也可以使用 docker 的默认 AppArmor 配置文件。docker 的默认 AppArmor 配置文件 docker-default 默认适用于运行中的容器，存放在/etc/apparmor.d/docker 目录下。<br>1. 如果你的 Linux 操作系统适用于 SELinux，遵循下列步骤：2. 设置 SELinux 状态。3. 设置 SELinux 策略。4. 为 docker 容器创建或导入 SELinux 策略模板。以守护进程模式带 SELinux 启动 docker，见下：docker daemon --selinux-enabled 使用如下安全选项启动 docker 容器，见下：docker run --interactive --tty --security-opt label=level:TopSecret centos /bin/bash |
| 5.7.2.2 | 执行以下命令，匹配 User=\s*.+且不含有 User=\s*root 如果为空，说明是 root 用户运行。<br>docker ps --quiet --all\| xargs docker inspect --format '{{ .Id }}: User={{.Config.User}}' 2>/dev/null \| egrep 'User=' | 确保容器镜像的 Dockerfile 包含以下指令：<br>USER <username or ID> |
| 5.7.2.3 | 执行以下命令，匹配 Not Empty 查询出容器的 Capability 能力并由用户来确认是否是需要。<br>docker ps --quiet --all\| xargs docker inspect --format '{{ .Id }}: CapAdd={{ .HostConfig.CapAdd }} CapDrop={{ .HostConfig.CapDrop }}' 2>/dev/null \| egrep 'CapAdd=.+\s+CapDrop=.+$' | 删除容器进程明确要求的 Capability 以外的所有 Capability |
| 5.7.2.4 | 执行以下命令，结果匹配 Privileged\s*=\s*false。<br>docker ps --quiet --all\| xargs docker inspect --format '{{ .Id }}: Privileged={{ .HostConfig.Privileged }}' 2>/dev/null | 限制特权容器的准入。在 k8s 1.21 以下版本，向集群中具有用户工作负载的每个命名空间添加 PSP 策略；在 K8S 1.21 以上版本建议开启 Pod Security Admission |

表 A.23（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.2.5 | 执行以下命令，结果不匹配 Source:(/\|/boot\|/dev\|/etc\|/lib\|/proc\|/sys\|/usr)\s+.*RW:true。<br>docker ps --quiet --all\| xargs docker inspect --format '{{ .Id }}: Volumes={{ .Mounts }}' 2>/dev/null | 尤其是在读写模式下。如果产品由于特殊场景必须 c 挂载，必须只以只读权限挂载 |
| 5.7.2.6 | 执行以下命令，结果不匹配(Ports=map\[(\d{1,3})/tcp)\|(Ports=map\[10[01]\d/tcp)\|(Ports=map\[102[0-3]/tcp))。<br>docker ps --quiet \| xargs docker inspect --format '{{ .Id }}: Ports={{ .NetworkSettings.Ports }}' 2>/dev/null | 在启动容器时不要将容器端口映射到特权主机端口上。另外，确保 dockerfile 文件里没有类似容器到主机特权端口的声明 |
| 5.7.2.7 | 执行以下命令，取得所有镜像打开的端口信息由用户确认是否是需要的。<br>docker ps --quiet \| xargs docker inspect --format '{{.Id}}：Ports = {{.NetworkSettings.Ports}}' | 修复容器镜像的 dockerfile，只暴露容器化应用程序所需的端口。启动容器时，不使用“-P”（大写）或“--publish-all”标志，即可完全忽略在 dockerfile 中定义的端口列表。使用“-p”（小写）或“--publish”标志，明确定义特定容器实例所需的端口。例如，docker run --interactive --tty --publish 5000 --publish 5001 --publish 5002 centos /bin/bash |
| 5.7.2.8 | 执行以下命令，结果不匹配 ReadonlyRootfs=false<br>docker ps --quiet --all\| xargs docker inspect --format '{{ .Id }}: ReadonlyRootfs={{ .HostConfig.ReadonlyRootfs }}' 2>/dev/null | 添加一个“--read-only”标志，使容器的根文件系统以只读方式挂载。可以与容器卷组合使用，强制容器的进程只写入将被持久化的位置。例如，docker run --interactive --tty --read-only --volume /centdata centos /bin/bash 这将运行具有只读根文件系统的容器，并将使用‘centdata’作为写操作的容器卷 |
| 5.7.2.9 | 执行以下命令，结果不匹配 NetworkMode=host<br>docker ps --quiet --all\| xargs docker inspect --format '{{ .Id }}: NetworkMode={{ .HostConfig.NetworkMode }}' 2>/dev/null | 限制 hostNetwork 容器的创建。<br>在 k8s 1.21 以下版本，向集群中具有用户工作负载的每个命名空间添加 PSP 策略；在 K8S 1.21 以上版本建议开启 Pod Security Admission |
| 5.7.2.10 | 执行以下命令，结果不匹配 IpcMode=host<br>docker ps --quiet --all\| xargs docker inspect --format '{{ .Id }}: IpcMode={{ .HostConfig.IpcMode }}' 2>/dev/null | 限制 hostIPC 容器的创建。在 k8s 1.21 以下版本，向集群中具有用户工作负载的每个命名空间添加 PSP 策略；在 K8S 1.21 以上版本建议开启 Pod Security Admission |

表 A. 23（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.2.11 | 执行以下命令，结果不匹配 PidMode=host<br>docker ps --quiet --all\| xargs docker inspect --format '{{ .Id }}: PidMode={{ .HostConfig.PidMode }}' 2>/dev/null | 限制 hostPID 容器的创建。在 k8s 1.21 以下版本，向集群中具有用户工作负载的每个命名空间添加 PSP 策略；在 K8S 1.21 以上版本建议开启 Pod Security Admission |
| 5.7.2.12 | 执行以下命令，判断结果是否为空，或符合最小的权限的需要<br>docker ps -q -a \|xargs docker inspect --format '{{.Id}}:Devices={{.HostConfig.Devices}}' 2>/dev/null | 不要将主机设备直接共享于容器。如果必须将主机设备共享给容器，使用正确的最小权限 |
| 5.7.2.13 | 列出集群中每个命名空间使用的控制策略，确保每个策略都不允许 NET_RAW 容器准入。<br>获取当前集群的 POD 名称，根据 POD 名称确认配置项 NET_RAW 是否被删除。<br>kubectl get pod <name> -o=jsonpath='{.spec.requiredDropCapabilities }' | 限制带有 NET_RAW 权限的容器的创建。在 k8s 1.21 以下版本，向集群中具有用户工作负载的每个命名空间添加 PSP 策略，确保 .spec.requiredDropCapabilities 包括 NET_RAW 或 ALL；在 K8S 1.21 以上版本建议开启 Pod Security Admission。 |
| 5.7.2.14 | 列出集群中每个命名空间使用的控制策略，确保每个策略都不允许使用 hostPath.Socket 的容器的准入。<br>以K8S配置项举例：<br>获取当前集群的POD名称，根据POD名称确认配置项是否存在volumes.hostPath，如存在则确认hostPath定义内容是否包含Socket。<br>kubectl get pod <name> -o=jsonpath='{.spec.hostPath}' | 确保严格控制对容器运行时套接字所在的文件系统访问。如果可能，限制为仅 root 用户可访问。<br>使用 Linux 内核命名空间等机制将 kubelet 与节点上运行的其他组件隔离。<br>确保限制或禁止使用包含容器运行时套接字的 hostPath 挂载， 无论是直接挂载还是通过挂载父目录挂载。此外，hostPath 挂载必须设置为只读，以降低攻击者绕过目录限制的风险。<br>限制用户对节点的访问，特别是限制超级用户对节点的访问。 |
| 5.7.2.15 | 列出集群中每个命名空间使用的控制策略，推荐每个策略都包含 RuntimeDefault 内容。<br>以 K8S 配置项举例：<br>获取当前集群的 POD 名称，根据 POD 名称确认配置项每个策略里是否使用了 seccompProfile，且类型为 type: RuntimeDefault。<br>kubectl get pod <name> -o=jsonpath='{.spec.seccompProfile}' | 1、对需要使用安全计算模式的容器，使用--seccomp-default 命令行标志来运行 kubelet<br>2、在容器配置文件中的 securityContext 部分增加:<br>spec:<br>securityContext:<br>seccompProfile:<br>type: RuntimeDefault |

表 A. 23（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.2.16 | 列出集群中每个命名空间使用的控制策略，确保容器策略都不允许未配置securityContext、capabilities、allowPrivilegeEscalation的容器准入。<br>以K8S配置项举例：<br>获取当前集群的POD名称，根据POD名称确认配置项securityContext、capabilities、allowPrivilegeEscalation是否符合权限最小化要求。<br>kubectl get pod <name> -o=jsonpath='{.spec.allowPrivilegeEscalation }'<br>kubectl get pod <name> -o=jsonpath='{.spec.runAsUser }'<br>kubectl get pod <name> -o=jsonpath='{.spec.capabilities }' | 通过配置 runAsUser，指定容器使用非 root 用户运行。<br>通过配置 capabilities，使用 capability 精确控制容器的特权访问权限。<br>通过配置 allowPrivilegeEscalation，在不需要容器进程提权的场景，建议关闭“允许特权逃逸”的配置。 |
| 5.7.2.17 | 获取当前集群的POD名称，根据POD名称确认配置项windowsOptions.hostProcess是否为false，如为True则确认其权限定义内容。<br>kubectl get pod <name> -o=jsonpath='{.spec.windowsOptions}' | 应限制使用 hostProcess 卷的容器的准入。在 k8s 1.21 以下版本，向集群中具有用户工作负载的每个命名空间添加 PSP 策略；在 K8S 1.21 以上版本建议开启 Pod Security Admission。 |
| 5.7.2.18 | 列出集群中每个命名空间使用的控制策略，确保策略中尽量避免允许使用HostPath卷策略的容器准入，在必须使用的情况下应检查HostPath卷定义的内容是否为满足运行的最小范围。<br>以K8S配置项举例：<br>获取当前集群的POD名称，根据POD名称确认配置项是否存在volumes.hostPath，如存在则确认hostPath定义内容。<br>kubectl get pod <name> -o=jsonpath='{.spec.hostPath}' | hostPath 卷可能会暴露特权系统凭据（例如 Kubelet）或特权 API（例如容器运行时套接字），可用于容器逃逸或攻击集群的其他部分。<br>应限制使用 hostPath 卷的容器的准入。在 k8s 1.21 以下版本，向集群中具有用户工作负载的每个命名空间添加 PSP 策略；在 K8S 1.21 以上版本建议开启 Pod Security Admission。 |
| 5.7.2.19 | 列出集群中每个命名空间使用的控制策略，确保策略中尽量对hostPorts进行限制，应检查hostPorts定义的端口范围是否为满足运行的最小范围。<br>以K8S配置项举例：<br>获取当前集群的POD名称，根据POD名称确认配置项是否存在hostPorts，如存在则确认hostPorts定义的端口范围。<br>kubectl get pod <name> -o=jsonpath='{.spec.hostPorts }' | 限制使用 HostPorts 的容器的准入。在 k8s 1.21 以下版本，向集群中具有用户工作负载的每个命名空间添加 PSP 策略；在 K8S 1.21 以上版本建议开启 Pod Security Admission。 |

### A.7.3 身份认证和访问控制

表 A. 24

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.3.1 | 执行如下命令，查询 pod 的使用账户：<br>kubectl get po -n example-namespace [podname] -o yaml \| grep serviceAccount<br>执行如下命令查看默认用户是否具有对应权限，如果返回为 false 则默认用户不含有对应权限，一般情况下默认用户不包含任何权限。 | 当 Kubernetes 工作负载需要特定访问 Kubernetes API 服务器时，需要创建显式的服务帐户。<br>修改每个默认服务帐户的配置以包含此值：automountServiceAccountToken: false |
| 5.7.3.2 | 执行如下命令查看 service account 属性<br>kubectl get sa -n test -o yaml<br>查看集群中的 Pod 和服务帐户对象，并确保设置了以下选项，除非资源明确要求此访问权限。<br>automountServiceAccountToken: false | 修改 Pod 定义和 service account 定义，禁止不必要的 token 挂载 |
| 5.7.3.3 | 检查 sc 所绑定的集群角色使用的权限，执行如下命令，可以看到该默认用户所使用的集群角色为如下:<br>kubectl get clusterrolebinding -n example-namespace -o yaml \| grep default -C5<br>roleRef<br>查看该集群角色所使用的权限见下：<br>kubectl get clusterrole -n example-namespace test -o yaml | 当 Kubernetes 工作负载需要特定访问 Kubernetes API 服务器时，需要创建显式的服务帐户,且按需分配最小化权限。 |
| 5.7.3.4 | 查看有权访问集群的所有凭据的列表，并确保未使用 system:masters 组 | 从集群中的所有用户中删除 system:masters 组 |
| 5.7.3.5 | 检查 sc 所绑定的集群角色使用的权限，执行如下命令，可以看到该默认用户所使用的集群角色为如下:<br>kubectl get clusterrolebinding -n example-namespace -o yaml \| grep default -C5<br>roleRef<br>查看该集群角色所使用的权限见下：<br>kubectl get clusterrole -n example-namespace test -o yaml<br><br>apiVersion: rbac.authorization.k8s.io/v1<br>kind: ClusterRole<br>metadata:<br>name: impersonator<br>rules:<br>- apiGroups: [""]<br>resources: ["users", "groups", "serviceaccounts"]<br>verbs: ["impersonate"] | 集群分配权限时最小化权限，谨慎分配 Bind、Impersonate 和 Escalate 权限。如无必要删除相关的 verb 和 clusterRole 。 |

表 A.24（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.3.6 | 通过查看有权访问 cluster-admin 角色的 ClusterRoleBindings 输出，获取有权访问 cluster-admin 角色的主体列表。查看列出的每个子项，确保它必须使用集群管理员权限。<br>kubectl get clusterrolebindings -o=custom-columns=NAME:.metadata.name,ROLE:.roleRef.name,SUBJECT:.subjects[*].name \|grep cluster-admin | 将所有 clusterrolebindings 标识到 cluster-admin 角色。检查是否使用了这些角色，以及它们是否需要此角色，或者它们是否可以使用权限较少的角色。在可能的情况下，首先将用户绑定到较低特权的角色，然后删除 clusterrolebinding 到 cluster-admin 角色：<br>kubectl delete clusterrolebinding [name]<br>默认只有 system:masters 关联了 cluster-admin 角色。 |
| 5.7.3.7 | 使用如下命令排查 roles 和 clusterroles 中的通配符：<br>检索集群中每个命名空间中定义的角色，并检查通配符（*）<br>kubectl get roles --all-namespaces -o yaml<br>检索集群中定义的集群角色并检查通配符（*）<br>kubectl get clusterroles -o yaml | 在可能的情况下，用特定的对象或操作替换 clusterroles 和 roles 中的任何通配符使用 |

A.7.4 防止信息泄露

表 A. 25

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.4.1 | 执行如下命令查看所有以环境变量形式使用 secret 的资源：<br>kubectl get all -o jsonpath='{range .items[?(@..secretKeyRef)]} {.kind} {.metadata.name} {"\n"}{end}' --all-namespaces | 如果可能的话，重写应用程序代码，从挂载的秘密文件中读取秘密，而不是从环境变量中读取 |
| 5.7.4.2 | 检查使用的机密管理实施方法 | 请参阅云提供商或第三方机密管理解决方案提供的机密管理选项 |
| 5.7.4.3 | 通过 kubectl 命令验证 secret 中敏感信息是否加密，对于较大的集群，可以通过命名空间对 secret 进行划分。<br>kubectl get secrets --all-namespaces | |
| 5.7.4.4 | 执行如下命令对私钥文件进行检查，确认私钥文件是否明文输出<br>cat /etc/kubernetes/pki/ca.key | 建议通过集成 KMC 秘钥库管理工具，在部署集群前使用派生的工作秘钥对私钥文件进行加密。集群各组件在读取私钥文件后，对私钥文件在内存中进行解密 |

表 A.25（续）

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.4.5 | 使用以下命令排查对 secret 拥有 get、list 或者 watch 权限的 role，由用户自己判断是否需要。<br>kubectl get roles --all-namespaces -o yaml<br>kubectl get clusterroles -o yaml | 在可能的情况下，删除对集群中 secret 对象的 get、list 和 watch 访问 |

### A.7.5 资源配额

表 A.26

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.5.1 | 执行如下命令检查每个 Pod 的 container 是否都有 CPU 资源请求(resources.requests.cpu)及资源限制(resources.limits.cpu)<br>kubectl get pods --all-namespaces | 为 Pod 内的每个 container 增加 CPU 资源请求(resources.requests.cpu)及资源限制(resources.limits.cpu) |
| 5.7.5.2 | 执行如下命令检查每个 Pod 的 container 是否都有 memory 资源请求(resources.requests.memory) 及资源限制(resources.limits.memory)<br>kubectl get pods --all-namespaces | 为 Pod 内的每个 container 增加 memory 资源请求(resources.requests.memory)及资源限制(resources.limits.memory) |
| 5.7.5.3 | 执行如下命令检查每个 namespaces 是否有设置配额<br>kubectl get ResourceQuota --all-namespaces | 按需为每个 namespace 设置资源配额，可控制的资源包括：CPU、内存、存储、pods、services、deployments、statefulsets 等，详细参考：https://kubernetes.io/zh-cn/docs/concepts/policy/resource-quotas/ |
| 5.7.5.4 | 执行如下命令检查每个 namespaces 是否有设置资源的限制范围<br>kubectl get LimitRange --all-namespaces | LimitRange 支持在一个命名空间中实施对每个 Pod 或 Container 最小和最大的资源使用量的限制，也可以在一个命名空间中实施对每个 PersistentVolumeClaim 能申请的最小和最大的存储空间大小的限制。请按需设置。 |
| 5.7.5.5 | 执行如下命令检查每个 service 是否有使用 nodePort 方式发布服务<br>kubectl get service --all-namespaces | 谨慎评估使用 NodePort 方式发布服务，若必须使用，建议定义单独的 NetworkPolicy，并结合使用 LoadBalancer 或 Ingress 来提高可用性和扩展性 |

### A.7.6 其他

表 A. 27

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.7.6.1 | 使用如下命令查看当前集群所有的命名空间：<br>kubectl get namespaces<br>并确保资源创建在期望的 namespace 下，确保这些命名空间是需要的且合理分配 | 按照 Kubernetes 文档，在需要时为部署中的对象创建命名空间。<br>不指定 namespace 时，资源默认创建在 default 命名空间下 |
| 5.7.6.2 | 使用下面命令列出默认命名空间中的对象：<br>kubectl get all<br>唯一的条目应该是系统管理的资源，如 kubernetes 服务 | 确保创建命名空间以允许 Kubernetes 资源的适当隔离，并且所有新资源都在特定命名空间中创建 |
| 5.7.6.3 | 通过以下方面来设置有效的 ClusterRoles 策略控制机制。<br>1. 最小化 ClusterRole 的使用<br>尽量使用 Role 而不是 ClusterRole,减少权限范围。只有需要跨命名空间访问的资源才使用 ClusterRole。<br>2. 避免过于宽泛的 ClusterRole<br>不要创建过于宽泛的 ClusterRole,访问权限应设置为所需最小权限。避免使用"*"权限。<br>3. 注意默认 ClusterRole<br>review 默认的 ClusterRole,如 view、edit、admin 等,移除不需要的权限控制。<br>4. 设置 ClusterRoleBinding 的服务帐号<br>使用 rbac.authorization.k8s.io/aggregator=true 注解,只允许集群管理员用户或服务账户绑定 ClusterRole。<br>5. 合理设置 RoleBinding 引用 review 哪些 RoleBinding 被设置为引用 ClusterRole,确保引用是必要的,防止权限范围扩大。<br>6. 定期审计 ClusterRole 和 Binding 开启审计日志,定期审计 ClusterRole 与 ClusterRoleBinding 的变更,发现风险权限提升情况。<br>7. 设置超时时间为 ClusterRoleBinding 设置时间限制,令牌定期更新,可以自动撤销已泄露的权限。 | 根据具体的需求设置合理的 ClusterRoles 策略控制机制，防止权限设置不当导致提权风险。 |

## A.8 CNI 插件和网络策略安全配置

### A.8.1 文件目录安全

表 A. 28

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.8.1.1 | 在 k8s master 节点运行如下命令进行检查，验证文件权限是否为 600 或者更严格，其中<path/to/cni/file>为容器网络接口文件的路径。<br>Stat -c %a <path/to/cni/files> | 如果不符合要求则对文件权限进行修改，可参考如下命令，在 master 节点上执行命令将文件权限修改为 600。<br>Chmod 600 <path/to/cni/files> |
| 5.8.1.2 | 在 k8s master 节点运行如下命令进行检查，验证文件属主是否为 root:root。其中<path/to/cni/file>为容器网络接口文件的路径。<br>Stat -c %U:%G <path/to/cni/files> | 如果不符合规则要求则修改文件属主，可参考如下命令，在 master 节点上执行命令修改文件属主为 root:root 。<br>chown root:root <path/to/cni/files> |

### A.8.2 网络策略安全配置

表 A. 29

| 要求 | 检查方法 | 修复方法 |
| --- | --- | --- |
| 5.8.2.1 | 查看集群使用的 CNI 插件的文档，并确认它支持入口和出口网络策略。 | 如果正在使用的 CNI 插件不支持网络策略，则应考虑使用不同的插件，或找到限制 Kubernetes 集群中流量的替代机制 |
| 5.8.2.2 | 通过以下方面来设置合理的 NetworkPolicy 资源，以确保 pod 之间通信安全，还应确保所有 NetworkPolicy 至少针对一个 Pod。<br>**1、默认拒绝所有流量**：这是一种保守但安全的策略，它可以防止任何未经授权的流量进入或离开你的集群。<br>**2、允许特定流量**：在默认拒绝所有流量的基础上，你可以根据你的业务需求来允许特定的流量。你可以通过指定 podSelector、namespaceSelector、ipBlock 等字段来定义允许的流量源或目标，以及通过指定 ports、protocol 等字段来定义允许的流量类型。<br>**3、隔离不同层次或角色的流量**：在一个复杂的集群中，你可能需要隔离不同层次或角色的流量，例如前端和后端、开发和生产、用户和管理员等。你可以通过使用标签 | 根据需要配置合理的 networkpolicy 策略 |

| | | |
|---|---|---|
| | 和选择器来实现这一点，给不同层次或角色的 Pod 添加不同的标签，并在网络策略中使用 podSelector 或 namespaceSelector 来选择它们。<br>**4、允许特定的外部流量**：在一些场景下，你可能需要允许你的 Pod 访问或被访问一些外部的资源，例如互联网、其他集群、其他网络等。你可以通过使用 ipBlock 字段来实现这一点，指定一个 CIDR 范围或一个 IP 地址来定义外部的流量源或目标 | |
| 5.8.2.3 | 通过以下命令查看所有命名空间下的 networkpolicy。<br>Kubectl get networkpolicy –all-namespaces<br>确保集群中定义的每个命名空间至少有一个 networkpolicy | 根据需要创建 networkpolicy 对象 |

# 附 录 B
# （资料性附录）
# 部分要求的场景限制

以下为正文 5.7.2 节部分要求的使用场景限制。

## B.1 特权容器使用场景

业务确实有使用特权容器的场景，在满足安全最小化原则，并充分做好安全风险评估前提下，允许以特权容器运行，典型场景包括：

- 业务需要容器内修改时间同步修改所在宿主机的时间；容器存储接口（CSI），需要将宿主机存储实例挂载到容器内部，需要获取宿主机文件访问权限，获取与主机一致的挂载信息；
- OS 升级，需要将虚拟机根目录挂载到容器的根目录上，因此容器内下载系统包、解压、安装的操作需要 root 权限；
- 组件需要访问宿主机相关资源来采集指标或驱动设置，包括宿主机网络信息，进程信息，文件信息（如/sys/、/proc/等目录下的信息），因此所在的容器需要能够访问主机网络命名空间、主机进程命名空间、主机系统敏感目录；
- 网络负载均衡器需要将外部网络接入到容器内部，完成网络接入和转发功能。

## B.2 共享主机使用场景

如果主机的 PID 命名空间与容器共享，则基本上允许容器中的进程查看主机系统上的所有进程。这打破了主机和容器之间的进程级隔离。

有权访问容器的人最终可以知道在主机系统上运行的所有进程，甚至可以从容器内杀死主机系统进程。因此，不要与容器共享主机的进程命名空间。

应至少定义一个准入控制策略，禁止容器共享主机 PID 命名空间。

① 业务确实有使用容器共享主机的进程命名空间的场景，在满足安全最小化原则，并充分做好安全风险评估前提下，允许使用容器共享主机的进程命名空间，典型场景包括：

- 组件需要访问宿主机相关资源来采集指标，包括宿主机网络信息，进程信息，文件信息（如/sys/、/proc/等目录下的信息），因此所在的容器需要能够访问主机网络命名空间、主机进程命名空间、主机系统敏感目录；

② 业务确实有使用容器共享主机的 IPC 命名空间的场景，在满足安全最小化原则，并充分做好安全风险评估前提下，允许容器共享主机的 IPC 命名空间，典型场景包括：

- 组件需要访问宿主机相关资源来采集指标，包括宿主机网络信息，进程信息，文件信息（如/sys/、/proc/等目录下的信息），因此所在的容器需要能够访问主机的 IPC 命名空间、主机网络命名空间、主机进程命名空间、主机系统敏感目录；

③ 业务确实有使用容器共享主机网络命名空间的场景，在满足安全最小化原则，并充分做好安全风险评估前提下，允许使用容器共享主机网络命名空间，典型场景包括：

组件需要访问宿主机相关资源来采集指标或设置网卡驱动，包括宿主机网络信息，进程信息，文件信息（如/sys/、/proc/等目录下的信息），因此所在的容器需要能够访问主机网络命名空间、主机进程命名空

间、主机系统敏感目录；
④ 业务确实有需要在容器内提权的场景，在满足权限最小化原则，并充分做好安全风险评估前提下，允许开启 allowPrivilegeEscalation，典型场景包括：
-因 root 权限最小化，导致稳态以普通用户权限运行，且需要瞬时提权至 root 权限的业务；
⑤ 业务确实有使用 NET_RAW 的场景，在满足安全最小化原则，并充分做好安全风险评估前提下，允许容器使用 NET_RAW 能力，典型场景包括：

- 网络负载均衡器需要将外部网络接入到容器内部，完成网络接入和转发功能；
- 容器需要实时修改宿主机 iptables 规则来保持 VIP 的时效性；

## 参 考 文 献

[1] CIS Kubernetes Benchmarks
[2] kubernetes官方服务文档 [DB/OL].[2023-07-27] https://kubernetes.io/docs/home/
[3] docker官方服务文档[DB/OL]. [2023-07-27].
https://docs.docker.com/get-started/overview/
[4] RFC 8446 [DB/OL]. [2023-07-27]. https://datatracker.ietf.org/doc/html/rfc8446