武汉大学国家网络安全学院

# 实 验 报 告

|  |  |
|---|---|
| 课 程 名 称： | 网络安全实验 |
| 实 验 名 称： | 防火墙实验 |
| 指 导 老 师： |  |
| 学 生 学 号： |  |
| 学 生 姓 名： |  |
| 完 成 日 期： | 2022.03.27 |

## 【实验描述】

从防火墙防护范围讲，可以大体分为主机防火墙和网络防火墙。主机防火墙：作用于单个机算机系统，例如个人电脑上 windows 自带的防火墙，linux 系统上的 iptables。网络防火墙：往往处于网络入口或边缘，对企业网络入口进行防护，服务范围为整个企业的内部网络。中小企业可以使用一台 x86 服务器运用 linux 的 iptables 搭建功能强大的网络防火墙。

网络防火墙和主机防火墙在企业网络架构中所处的网络位置不一样，它俩的结合能使企业网络更安全。

从产品形态讲，防火墙可以分为硬件防火墙和软件防火墙。软件防火墙 iptables，其实它并不是真正的防火墙，我们可以理解它为一个命令行工具，位于用户空间，我们用这个工具操作“安全框架”。netfilter 是防火墙真正的安全框架（framework），netfilter 位于内核空间。

Iptables 按照规则（rules）来处理数据包，规则其实就是预定义的条件。规则存储在内核空间的信息包过滤表中，这些规则分别指定了源地址、目的地址、源端口、目的端口、协议等。当数据包与规则匹配时，iptables 就根据规则所定义的方法来处理这些数据包，如放行（accept）、拒绝（reject）和丢弃（drop）等。配置防火墙的主要工作就是添加、修改和删除这些规则。

防火墙实验旨在通过配置 iptables 规则的应用实战，让学生深刻了解掌握主机安全、网络安全的防护策略。结合网络虚拟化技术，了解和掌握云计算中部分网络虚拟化功能的实验。

本实验内容共包含 6 个子任务，分别是：

任务一 了解 Iptables 防火墙基础；

任务二 掌握 Iptables 主机防火墙常用配置和审计功能；

任务三 了解虚拟网络中常用软件；

任务四 利用虚拟网络构建网络防火墙实验场景；

任务五 使用 Iptables 实战网络防火墙 NAT、网络控制；

任务六 Iptables 实战公有云二层防火墙。

## 【实验目的】

了解 Linux 防火墙 iptables 的基础概念，掌握其基本命令，会进行简单的规则的增删查。

掌握 IPtables 工作中的常用策略。

掌握 iptables 的审计策略。

通过 Iptables 工作中的常用策略以及审计策略的学习和使用，具备丰富的主机安全防护的能力。

了解 openswitch 与 namespace 的概念。

掌控 Linux 虚拟网络设备的使用。

掌握 SDN 交换机 openvswitch 的安装与使用。

掌握利用 openswitch 与 netns 来搭建虚拟网络环境。

掌握 iptables nat 的基础知识与常用配置的操作。

掌握简单的网络防火墙的访问控制操作。

掌握企业公有云中二层防火墙实现的一种方法。

通过以上工具和知识的学习和使用，能够融会贯通，掌握 iptables 防火墙配置的原理和方法，掌握自主学习和实践企业网络防火墙的搭建与配置策略，使所处工作环境处于安全的状态。

## 【实验环境】

| 操作系统 | IP地址 | 服务器角色 | 登录账户密码 |
|---|---|---|---|
| centos7 | 192.168.1.11 | 操作机 | 用户名：root；密码：Simplexue123 |

## 【实验工具】

Xshell 7
openvswitch
ovs-vsctl (openvswitch 软件提供)
iptables
brctl
ip (包含 ip link、ip netns、ip addr 等)
tcpdump
ethtool

route
sysctl
modprobe
ipset

## 【实验步骤】

任务一

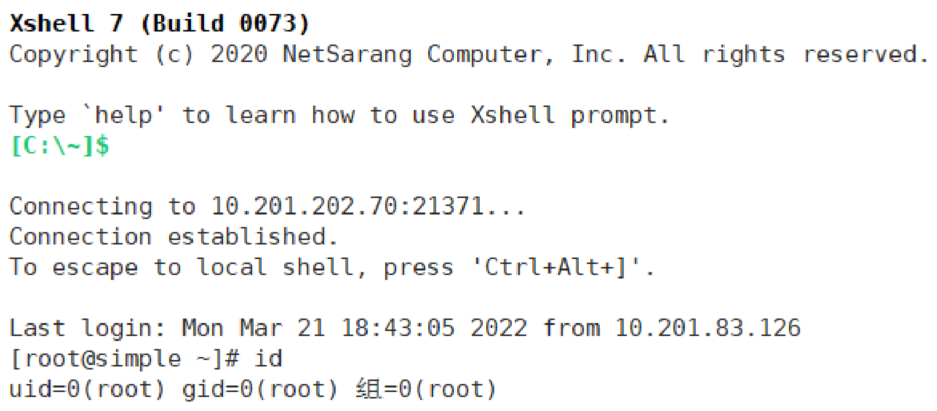
### 1.1 防火墙一些概念

### 1.2 Iptables 基础

### 1.3 Iptables 使用语法

### 1.4 登陆实验机

使用 Xshell 7 登陆实验机，用户名是 root，密码是 Simplexue123。

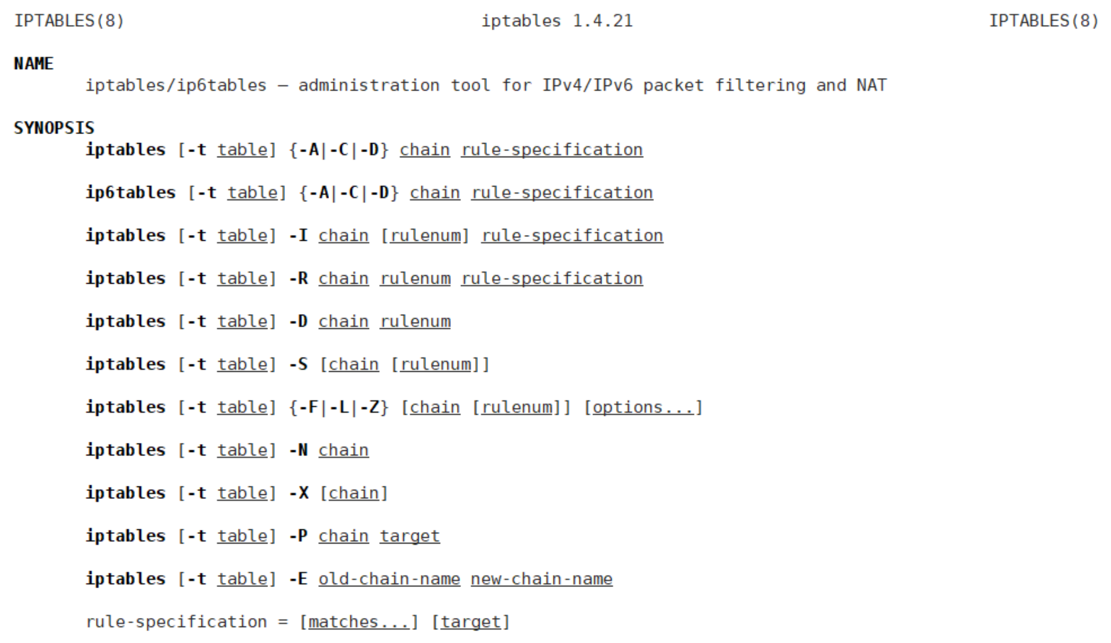


```
Xshell 7 (Build 0073)
Copyright (c) 2020 NetSarang Computer, Inc. All rights reserved.

Type `help' to learn how to use Xshell prompt.
[C:\~]$

Connecting to 10.201.202.70:21371...
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.

Last login: Mon Mar 21 18:43:05 2022 from 10.201.83.126
[root@simple ~]# id
uid=0(root) gid=0(root) 组=0(root)
```

### 1.5 Iptables 简单使用

查看 iptables 使用手册。

man iptables



```
IPTABLES(8)                      iptables 1.4.21                      IPTABLES(8)

NAME
       iptables/ip6tables — administration tool for IPv4/IPv6 packet filtering and NAT

SYNOPSIS
       iptables [-t table] {-A|-C|-D} chain rule-specification

       ip6tables [-t table] {-A|-C|-D} chain rule-specification

       iptables [-t table] -I chain [rulenum] rule-specification

       iptables [-t table] -R chain rulenum rule-specification

       iptables [-t table] -D chain rulenum

       iptables [-t table] -S [chain [rulenum]]

       iptables [-t table] {-F|-L|-Z} [chain [rulenum]] [options...]

       iptables [-t table] -N chain

       iptables [-t table] -X [chain]

       iptables [-t table] -P chain target

       iptables [-t table] -E old-chain-name new-chain-name

       rule-specification = [matches...] [target]
```

查看 filter 表规则。iptables 命令的-t 参数可以指定查看哪个表的规则，默认查看 filter 表规则。因此以下两句命令效果一样。

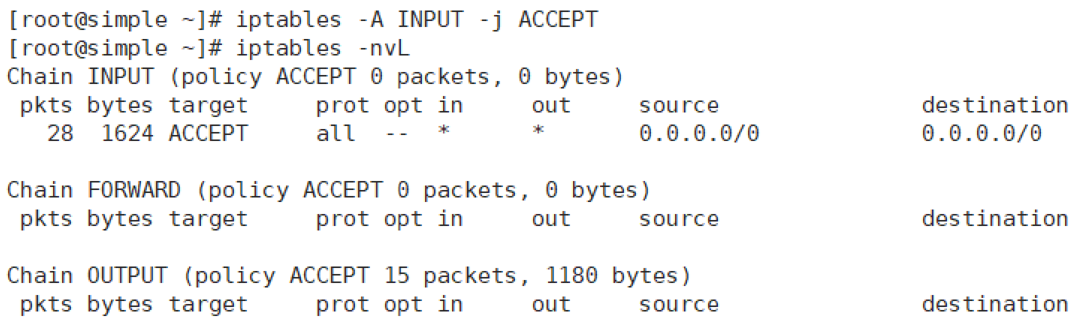
iptables -t filter -nvL

iptables -nvL

```
[root@simple ~]# iptables -t filter -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 29 packets, 1664 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 16 packets, 1680 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

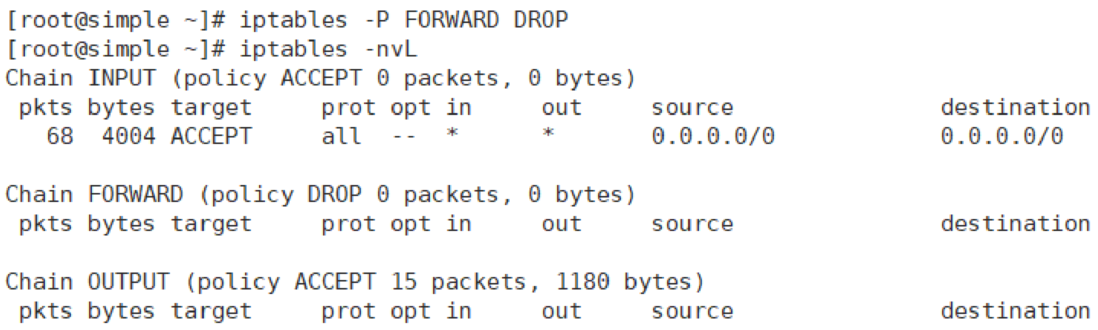
添加规则，允许所有网络访问本机。

iptables -A INPUT -j ACCEPT

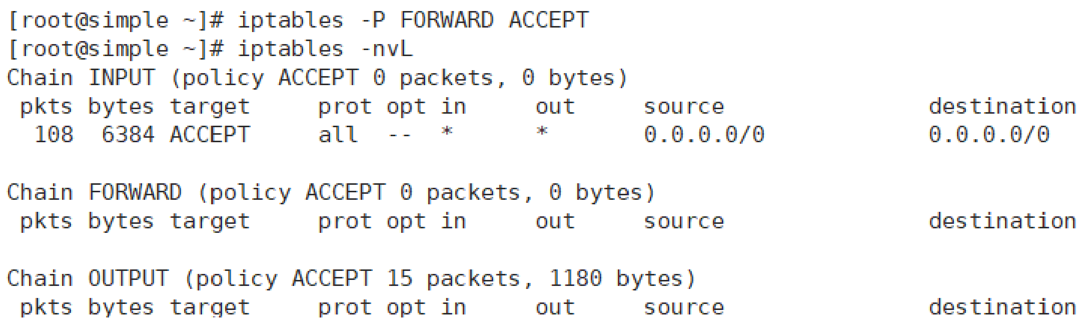

```
[root@simple ~]# iptables -A INPUT -j ACCEPT
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   28  1624 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

为 FORWARD 链添加默认规则。

iptables -P FORWARD DROP

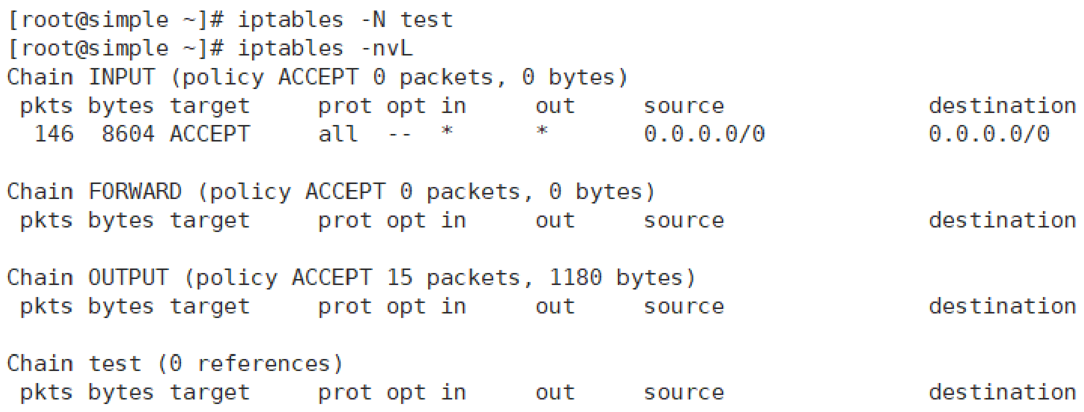

```
[root@simple ~]# iptables -P FORWARD DROP
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   68  4004 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

iptables -P FORWARD ACCEPT


```
[root@simple ~]# iptables -P FORWARD ACCEPT
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  108  6384 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

使用-N 参数添加自定链 test。

iptables -N test


```
[root@simple ~]# iptables -N test
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  146  8604 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain test (0 references)
 pkts bytes target     prot opt in     out     source               destination
```

使用-X 参数删除自定链 test。

iptables -X test

```
[root@simple ~]# iptables -X test
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  190 11136 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

## 任务二

### 2.1 常用语法

（1）匹配指定协议。

-p，--protocol

iptables -A INPUT -p tcp -j ACCEPT

```
[root@simple ~]# iptables -A INPUT -p tcp -j ACCEPT
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   28  1624 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

（2）以 IP 源地址匹配包。

-s，--src，--source

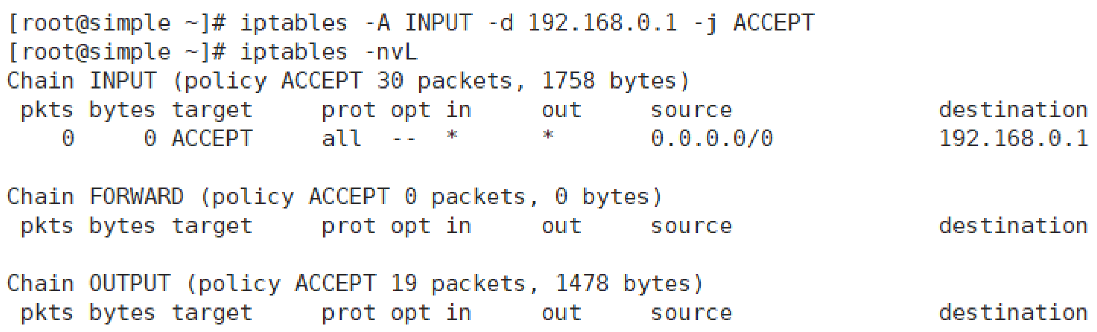
iptables -A INPUT -s 192.168.0.1 -j ACCEPT

```
[root@simple ~]# iptables -A INPUT -s 192.168.0.1 -j ACCEPT
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       192.168.0.1          0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

（3）以 IP 目的地址匹配包。

-d，--dst，--destination

iptables -A INPUT -d 192.168.0.1 -j ACCEPT

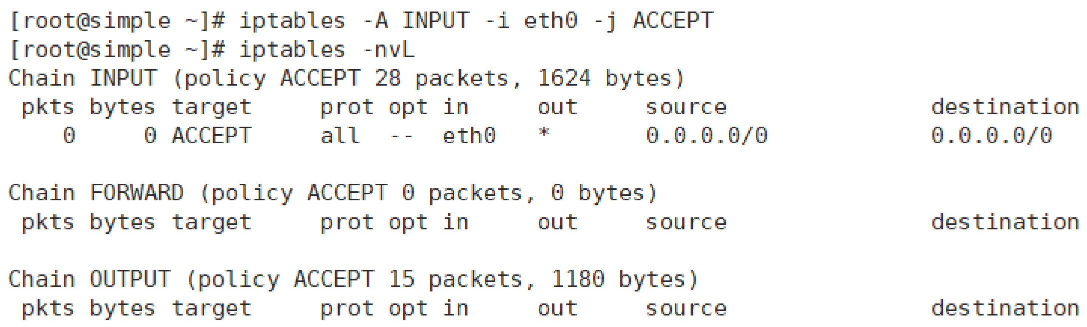


```
[root@simple ~]# iptables -A INPUT -d 192.168.0.1 -j ACCEPT
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 30 packets, 1758 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            192.168.0.1

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 19 packets, 1478 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

（4）以包进入本地使用的网络接口匹配包。

-i

iptables -A INPUT -i eth0 -j ACCEPT

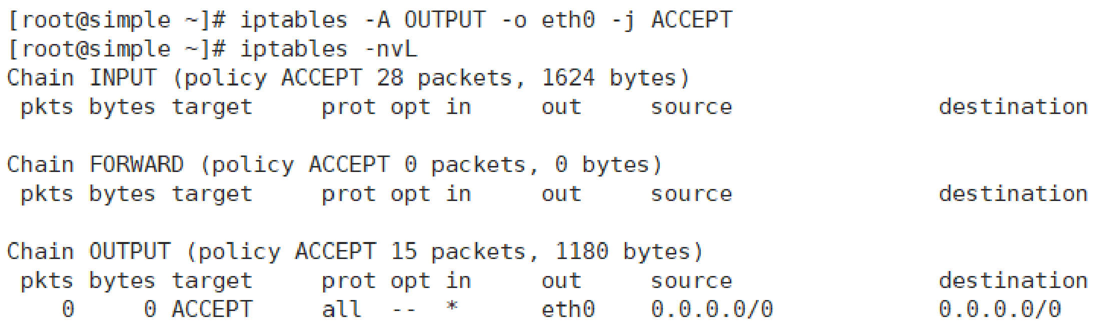


```
[root@simple ~]# iptables -A INPUT -i eth0 -j ACCEPT
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  eth0   *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

（5）以包离开本地所使用的网络接口来匹配包。

-o

iptables -A OUTPUT -o eth0 -j ACCEPT

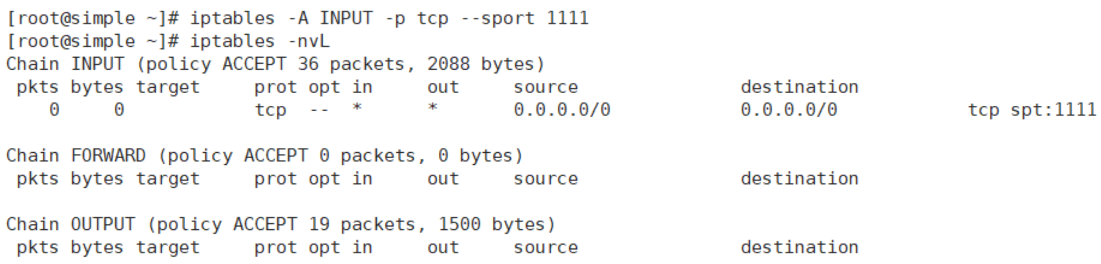


```
[root@simple ~]# iptables -A OUTPUT -o eth0 -j ACCEPT
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      eth0    0.0.0.0/0            0.0.0.0/0
```

（6）匹配通信源端口。

--source-port，--sport

iptables -A INPUT -p tcp --sport 1111

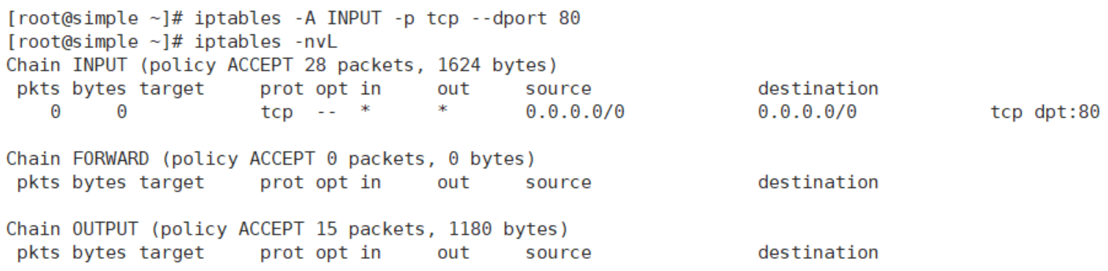


```
[root@simple ~]# iptables -A INPUT -p tcp --sport 1111
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 36 packets, 2088 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0            tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp spt:1111

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 19 packets, 1500 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

（7）匹配通信目的端口。

-- destination-port，--dport

iptables -A INPUT -p tcp --dport 80

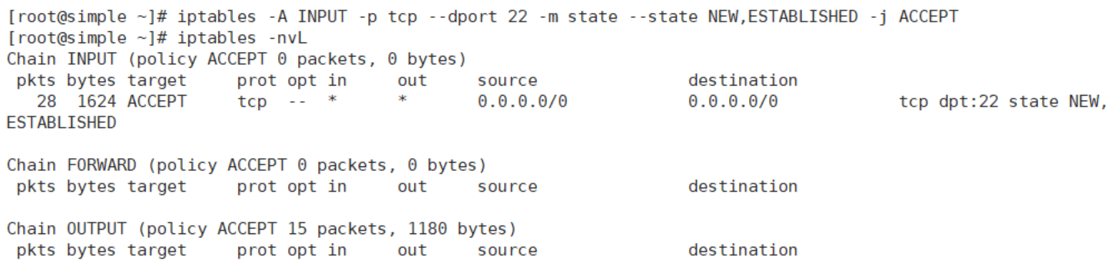


```
[root@simple ~]# iptables -A INPUT -p tcp --dport 80
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination         
    0     0            tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:80

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination         

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination         
```

## 2.2 状态检测

-m state --state {NEW,ESTATBLISHED,INVALID,RELATED}

iptables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT

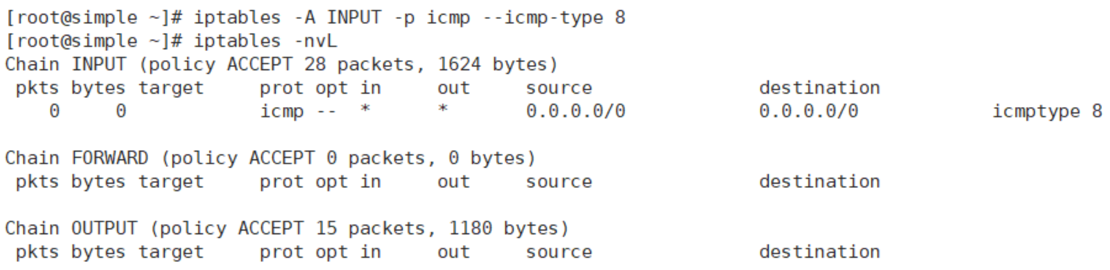


```
[root@simple ~]# iptables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination         
   28  1624 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:22 state NEW,
ESTABLISHED

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination         

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination         
```

## 2.3 特殊参数

--icmp-type  指定 ICMP 的类型编号

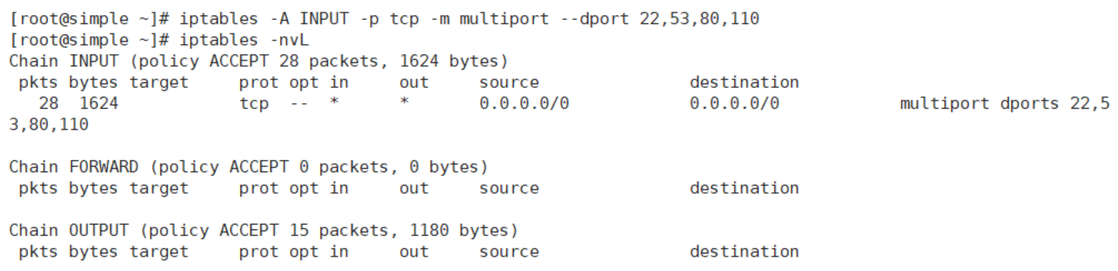
iptables -A INPUT -p icmp --icmp-type 8



```
[root@simple ~]# iptables -A INPUT -p icmp --icmp-type 8
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination         
    0     0            icmp --  *      *       0.0.0.0/0            0.0.0.0/0            icmptype 8

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination         

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination         
```

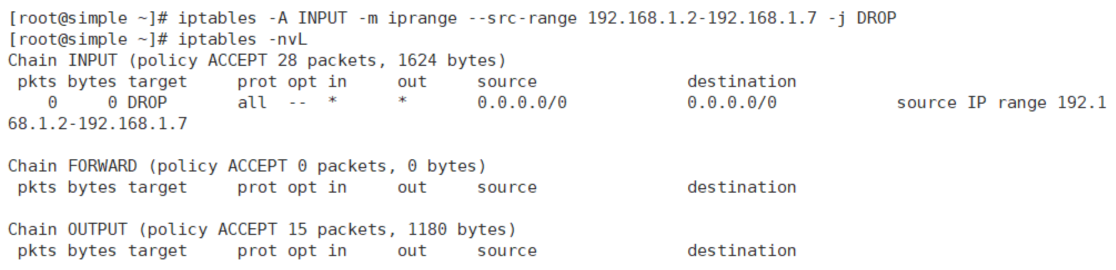
-m multiport  指定多端口号

iptables -A INPUT -p tcp -m multiport --dport 22,53,80,110


```
[root@simple ~]# iptables -A INPUT -p tcp -m multiport --dport 22,53,80,110
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination
   28  1624            tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           multiport dports 22,5
3,80,110

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

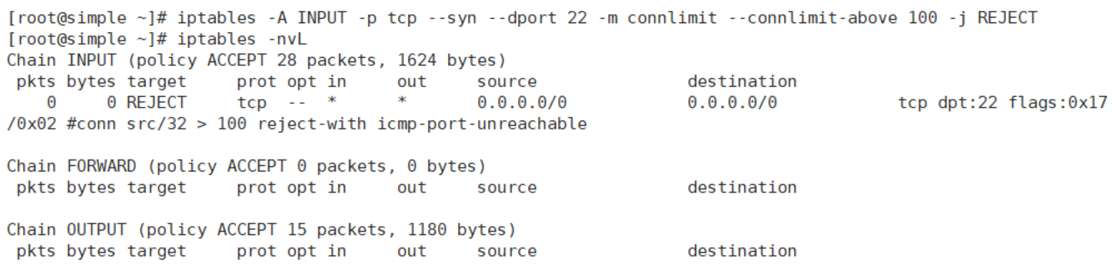
-m iprange 指定 IP 段

iptables -A INPUT -m iprange --src-range 192.168.1.2-192.168.1.7 -j DROP


```
[root@simple ~]# iptables -A INPUT -m iprange --src-range 192.168.1.2-192.168.1.7 -j DROP
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0           source IP range 192.1
68.1.2-192.168.1.7

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

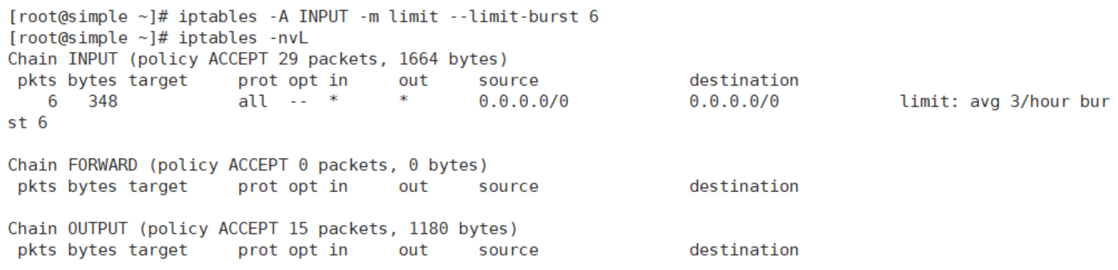
-m connlimit 连接限定

iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 100 -j REJECT


```
[root@simple ~]# iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 100 -j REJECT
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REJECT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22 flags:0x17
/0x02 #conn src/32 > 100 reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

-m limit 限定连接速率，也就是限定匹配数据包的个数

iptables -A INPUT -m limit --limit-burst 6

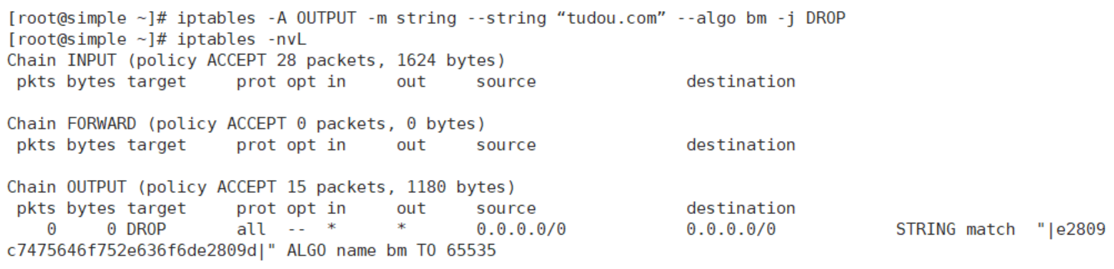

```
[root@simple ~]# iptables -A INPUT -m limit --limit-burst 6
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 29 packets, 1664 bytes)
 pkts bytes target     prot opt in     out     source               destination
    6   348            all  --  *      *       0.0.0.0/0            0.0.0.0/0           limit: avg 3/hour bur
st 6

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

-m string 按字符串限定

--algo bm|kmp # 指定算法 bm 或 kmp

--string "STRING" # 指定字符串本身

iptables -A OUTPUT -m string --string "tudou.com" --algo bm -j DROP


```
[root@simple ~]# iptables -A OUTPUT -m string --string “tudou.com” --algo bm -j DROP
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            STRING match  "|e2809
c7475646f752e636f6de2809d|" ALGO name bm TO 65535
```

## 2.4 iptables 常用的操作语法

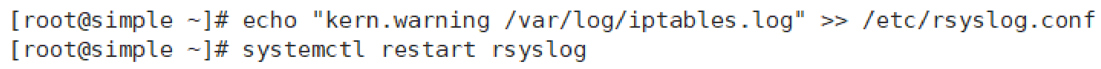
## 2.5 iptables 日志记录

往/etc/syslog.conf 文件中加入一行内容。

echo "kern.warning /var/log/iptables.log" >> /etc/rsyslog.conf

然后重启 rsyslog 服务。

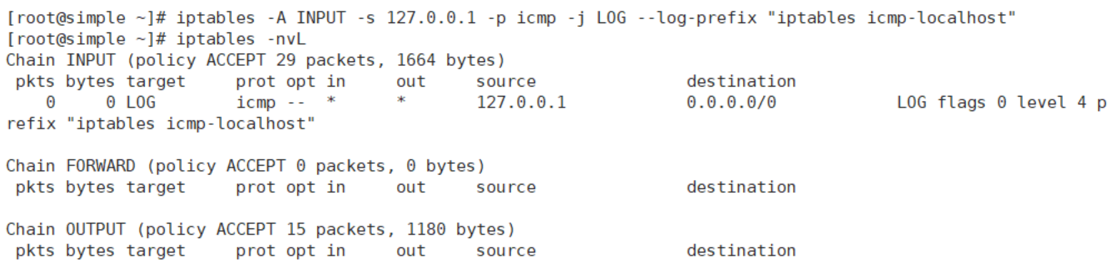
systemctl restart rsyslog


```
[root@simple ~]# echo "kern.warning /var/log/iptables.log" >> /etc/rsyslog.conf
[root@simple ~]# systemctl restart rsyslog
```

例：记录源为 127.0.0.1 的所有 ICMP 日志

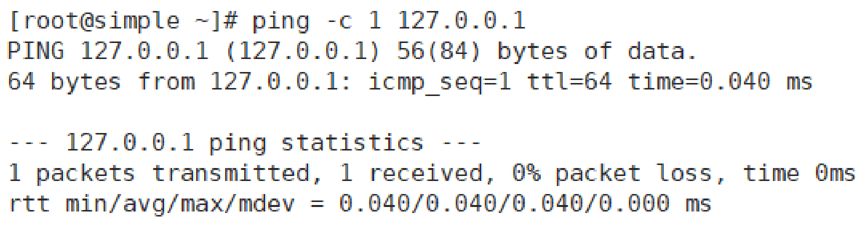
(1) 配置 iptables 策略

iptables -A INPUT -s 127.0.0.1 -p icmp -j LOG --log-prefix "iptables icmp-localhost"


```
[root@simple ~]# iptables -A INPUT -s 127.0.0.1 -p icmp -j LOG --log-prefix "iptables icmp-localhost"
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 29 packets, 1664 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 LOG        icmp --  *      *       127.0.0.1            0.0.0.0/0            LOG flags 0 level 4 p
refix "iptables icmp-localhost"

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

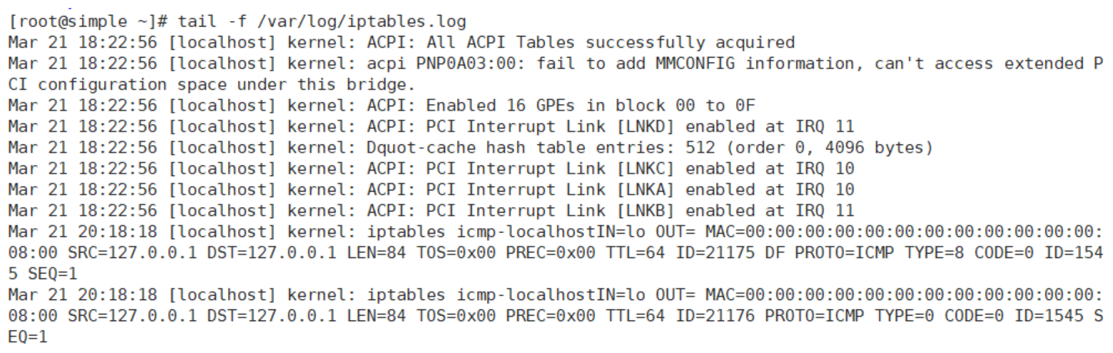
(2) 验证规则

ping -c 1 127.0.0.1


```
[root@simple ~]# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.040 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.040/0.040/0.040/0.000 ms
```

(3) 查看日志

tail -f /var/log/iptables.log



```
[root@simple ~]# tail -f /var/log/iptables.log
Mar 21 18:22:56 [localhost] kernel: ACPI: All ACPI Tables successfully acquired
Mar 21 18:22:56 [localhost] kernel: acpi PNP0A03:00: fail to add MMCONFIG information, can't access extended P
CI configuration space under this bridge.
Mar 21 18:22:56 [localhost] kernel: ACPI: Enabled 16 GPEs in block 00 to 0F
Mar 21 18:22:56 [localhost] kernel: ACPI: PCI Interrupt Link [LNKD] enabled at IRQ 11
Mar 21 18:22:56 [localhost] kernel: Dquot-cache hash table entries: 512 (order 0, 4096 bytes)
Mar 21 18:22:56 [localhost] kernel: ACPI: PCI Interrupt Link [LNKC] enabled at IRQ 10
Mar 21 18:22:56 [localhost] kernel: ACPI: PCI Interrupt Link [LNKA] enabled at IRQ 10
Mar 21 18:22:56 [localhost] kernel: ACPI: PCI Interrupt Link [LNKB] enabled at IRQ 11
Mar 21 20:18:18 [localhost] kernel: iptables icmp-localhostIN=lo OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:00:
08:00 SRC=127.0.0.1 DST=127.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=21175 DF PROTO=ICMP TYPE=8 CODE=0 ID=154
5 SEQ=1
Mar 21 20:18:18 [localhost] kernel: iptables icmp-localhostIN=lo OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:00:
08:00 SRC=127.0.0.1 DST=127.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=21176 PROTO=ICMP TYPE=0 CODE=0 ID=1545 S
EQ=1
```

(4) 上例 iptables 日志字段解释(编号 21 后为未用到的字段)

Mar 21 20:18:18 [localhost] kernel: iptables icmp-localhostIN=lo OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=127.0.0.1 DST=127.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=21175 DF PROTO=ICMP TYPE=8 CODE=0 ID=1545 SEQ=1

(5) 日志策略

可以配置日志策略获取所有 TCP、UDP、ssh 的日志。

iptables -A INPUT -p tcp -j LOG --log-prefix "iptables TCP "

iptables -A INPUT -p udp -j LOG --log-prefix "iptables UDP "

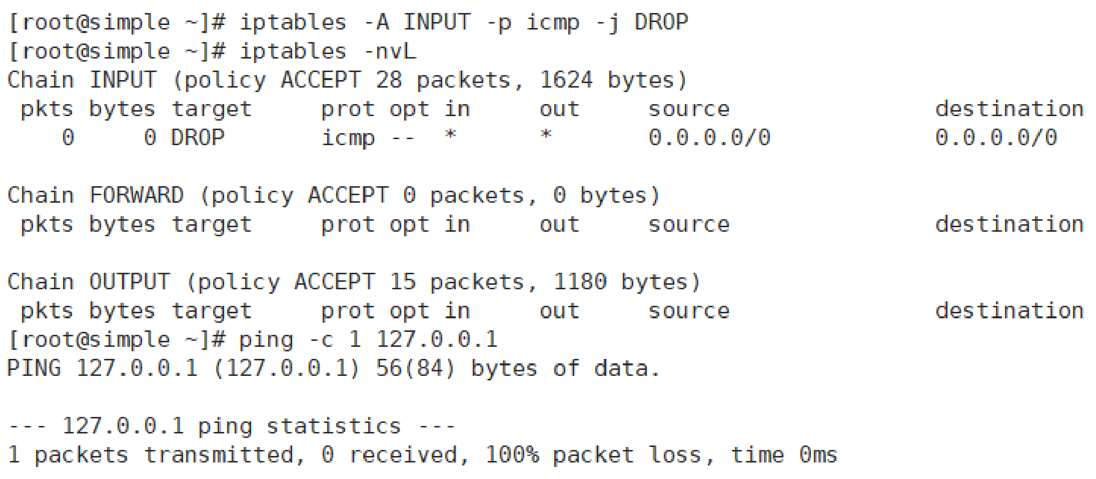
iptables -A INPUT -p tcp --dport 22 -j LOG --log-prefix "iptables SSH "

**2.6 自定义策略**

(1)禁止 ping 127.0.0.1

默认 icmp 数据包是 accept 的，可以自定义策略以禁止 ping 127.0.0.1。配置策略后发现已经无法 ping 通 127.0.0.1。

iptables -A INPUT -p icmp -j DROP

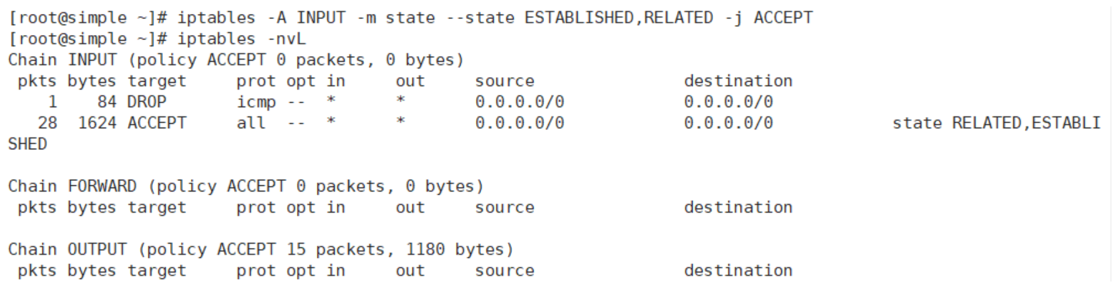


```
[root@simple ~]# iptables -A INPUT -p icmp -j DROP
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       icmp --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
[root@simple ~]# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

(2)  状态为已连接的放行

为已连接的状态放行。

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

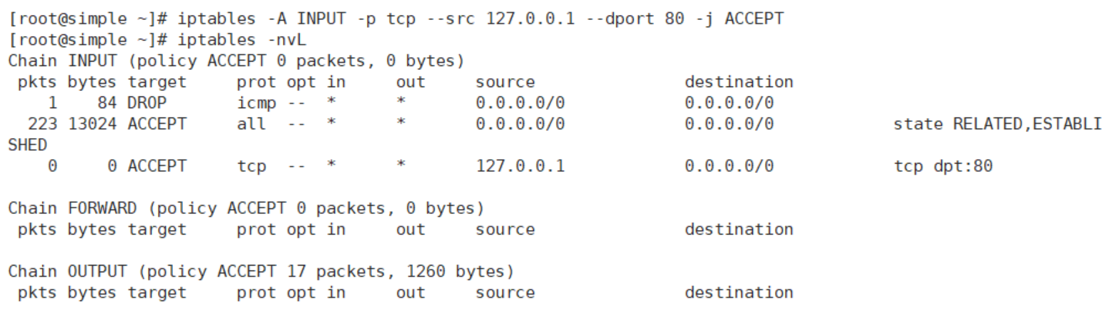


```
[root@simple ~]# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1    84 DROP       icmp --  *      *       0.0.0.0/0            0.0.0.0/0
   28  1624 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            state RELATED,ESTABLI
SHED

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

(3)只允许本机访问 80

只允许本机，即源地址 127.0.0.1 访问 80 目的端口。

iptables -A INPUT -p tcp --src 127.0.0.1 --dport 80 -j ACCEPT

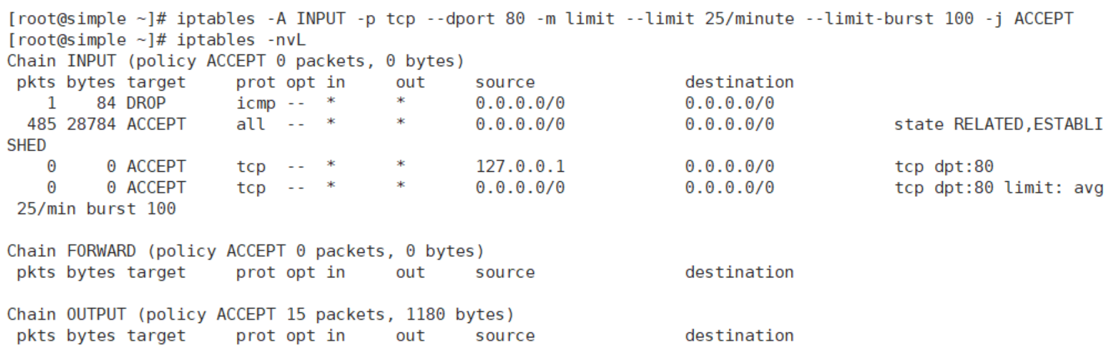


```
[root@simple ~]# iptables -A INPUT -p tcp --src 127.0.0.1 --dport 80 -j ACCEPT
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1    84 DROP       icmp --  *      *       0.0.0.0/0            0.0.0.0/0
  223 13024 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            state RELATED,ESTABLI
SHED
    0     0 ACCEPT     tcp  --  *      *       127.0.0.1            0.0.0.0/0            tcp dpt:80

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 17 packets, 1260 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

(4)  利用扩展模块 limit，可以实现 DoS 攻击防范

利用扩展模块 limit，可以限制每分钟允许通行的最大数据包数量。

iptables -A INPUT -p tcp --dport 80 -m limit --limit 25/minute --limit-burst 100 -j ACCEPT



```
[root@simple ~]# iptables -A INPUT -p tcp --dport 80 -m limit --limit 25/minute --limit-burst 100 -j ACCEPT
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1    84 DROP       icmp --  *      *       0.0.0.0/0            0.0.0.0/0
  485 28784 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            state RELATED,ESTABLI
SHED
    0     0 ACCEPT     tcp  --  *      *       127.0.0.1            0.0.0.0/0            tcp dpt:80
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:80 limit: avg
 25/min burst 100

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

任务三

**3.1 概述**

（1）Open vSwitch

Open vSwitch（简称为 OVS）是由 Nicira Networks 主导的，运行在虚拟化平台（例如 KVM，Xen）上的虚拟交换机。在虚拟化平台上，OVS 可以为动态变化的端点提供 2 层交换功能，很好的控制虚拟网络中的访问策略、网络隔离、流量监控等等。

OVS 遵循 Apache 2.0 许可证, 能同时支持多种标准的管理接口和协议。OVS 也提供了对 OpenFlow 协议的支持，用户可以使用任何支持 OpenFlow 协议的控制器对 OVS 进行远程管理控制。

（2）网络名称空间 netns

netns 是在 linux 中提供网络虚拟化的一个项目，使用 netns 网络空间虚拟化可以在本地虚拟化出多个网络环境。netns 可以让一台机器上模拟多个网络设备，是网络虚拟化的重要组成，将不同类型的网络应用隔离。

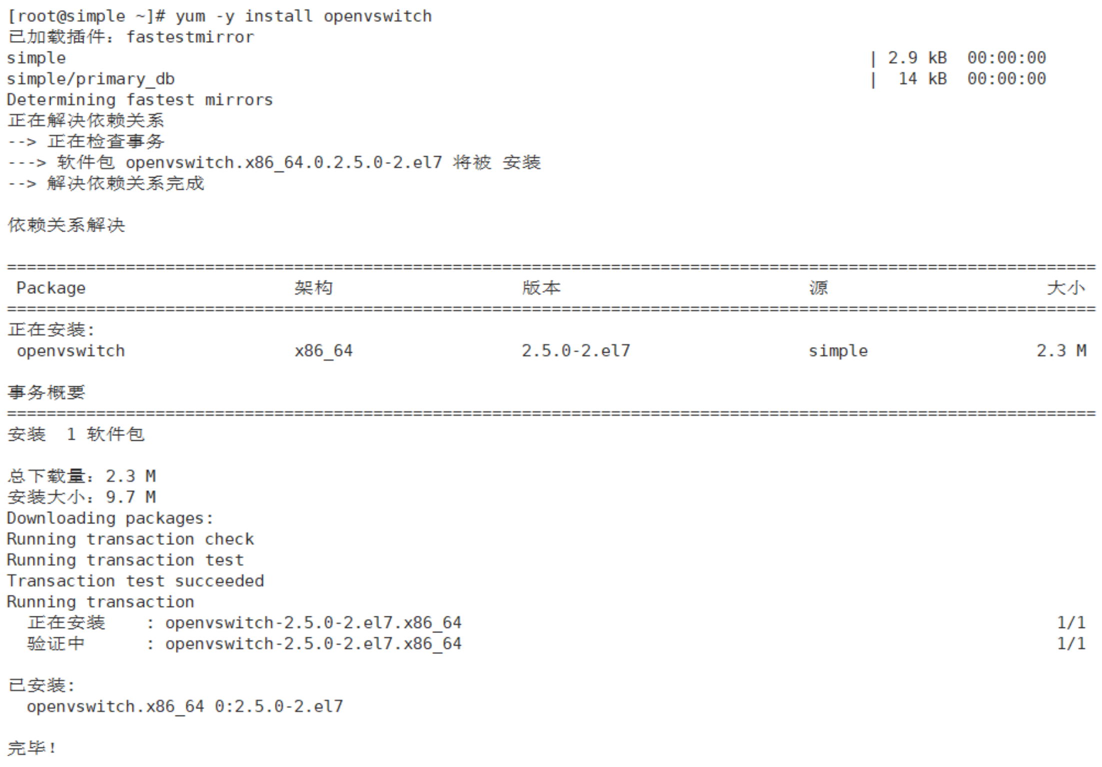
一个 net namespace 拥有独立的独立的网卡空间，路由表，ARP 表，ip 地址表，iptables 等。

**3.2 软件安装**

（1）netns 功能由系统网络配置工具 iproute2 提供，命令形式为 ip netns

（2）openvswitch 安装

```
yum -y install openvswitch
```



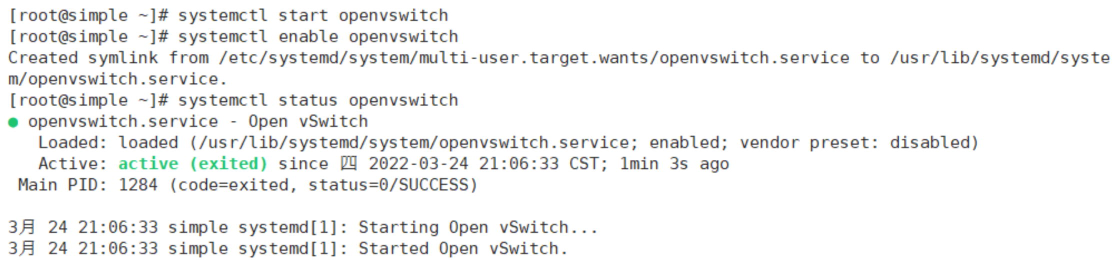
```
[root@simple ~]# yum -y install openvswitch
已加载插件：fastestmirror
simple                                                                          | 2.9 kB  00:00:00
simple/primary_db                                                               |  14 kB  00:00:00
Determining fastest mirrors
正在解决依赖关系
--> 正在检查事务
---> 软件包 openvswitch.x86_64.0.2.5.0-2.el7 将被 安装
--> 解决依赖关系完成

依赖关系解决

==============================================================================================================
 Package                     架构                    版本                         源                      大小
==============================================================================================================
正在安装:
 openvswitch                 x86_64                  2.5.0-2.el7                  simple                 2.3 M

事务概要
==============================================================================================================
安装  1 软件包

总下载量：2.3 M
安装大小：9.7 M
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  正在安装    : openvswitch-2.5.0-2.el7.x86_64                                                               1/1
  验证中      : openvswitch-2.5.0-2.el7.x86_64                                                               1/1

已安装:
  openvswitch.x86_64 0:2.5.0-2.el7

完毕！
```

(3) 启动 openvswitch 并设置自启

启动 openvswitch 并设置自启分别使用以下两条命令。

systemctl start openvswitch

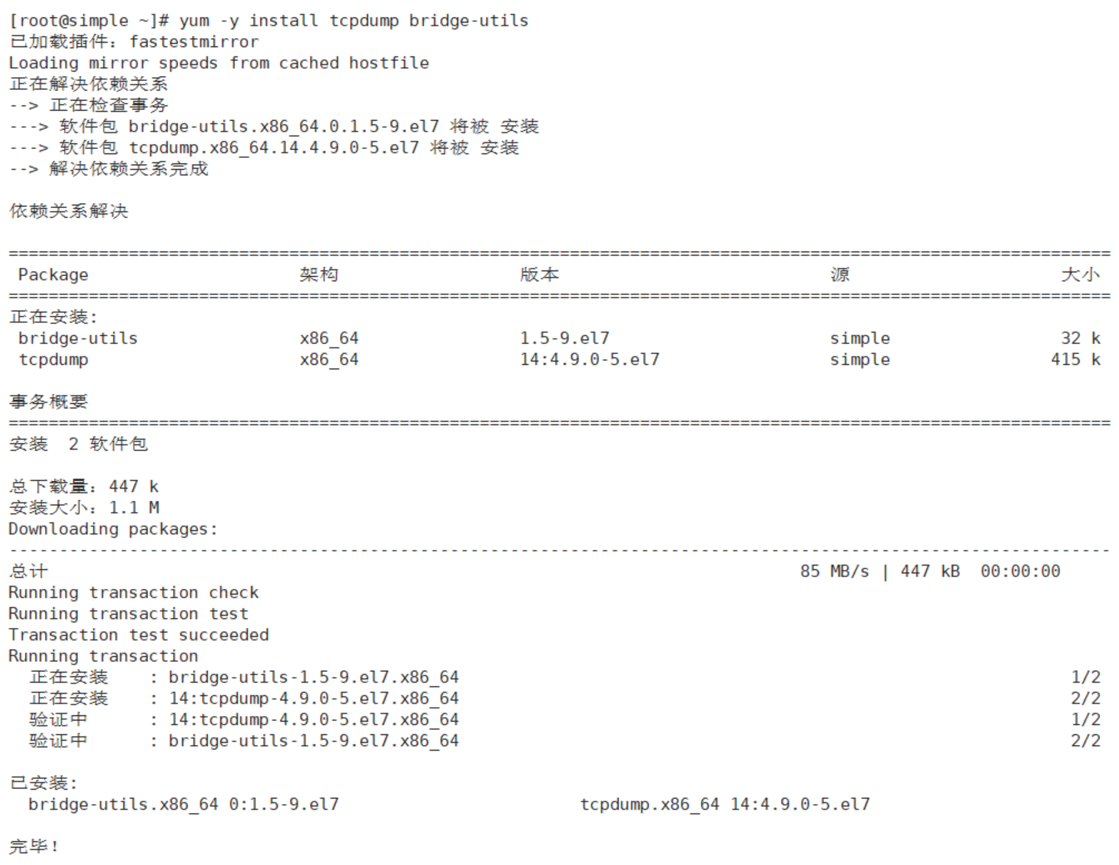
systemctl enable openvswitch



```
[root@simple ~]# systemctl start openvswitch
[root@simple ~]# systemctl enable openvswitch
Created symlink from /etc/systemd/system/multi-user.target.wants/openvswitch.service to /usr/lib/systemd/syste
m/openvswitch.service.
[root@simple ~]# systemctl status openvswitch
● openvswitch.service - Open vSwitch
   Loaded: loaded (/usr/lib/systemd/system/openvswitch.service; enabled; vendor preset: disabled)
   Active: active (exited) since 四 2022-03-24 21:06:33 CST; 1min 3s ago
 Main PID: 1284 (code=exited, status=0/SUCCESS)

3月 24 21:06:33 simple systemd[1]: Starting Open vSwitch...
3月 24 21:06:33 simple systemd[1]: Started Open vSwitch.
```

(4) 安装抓包工具 tcpdump,网络配置工具 bridge-utils

yum -y install tcpdump bridge-utils

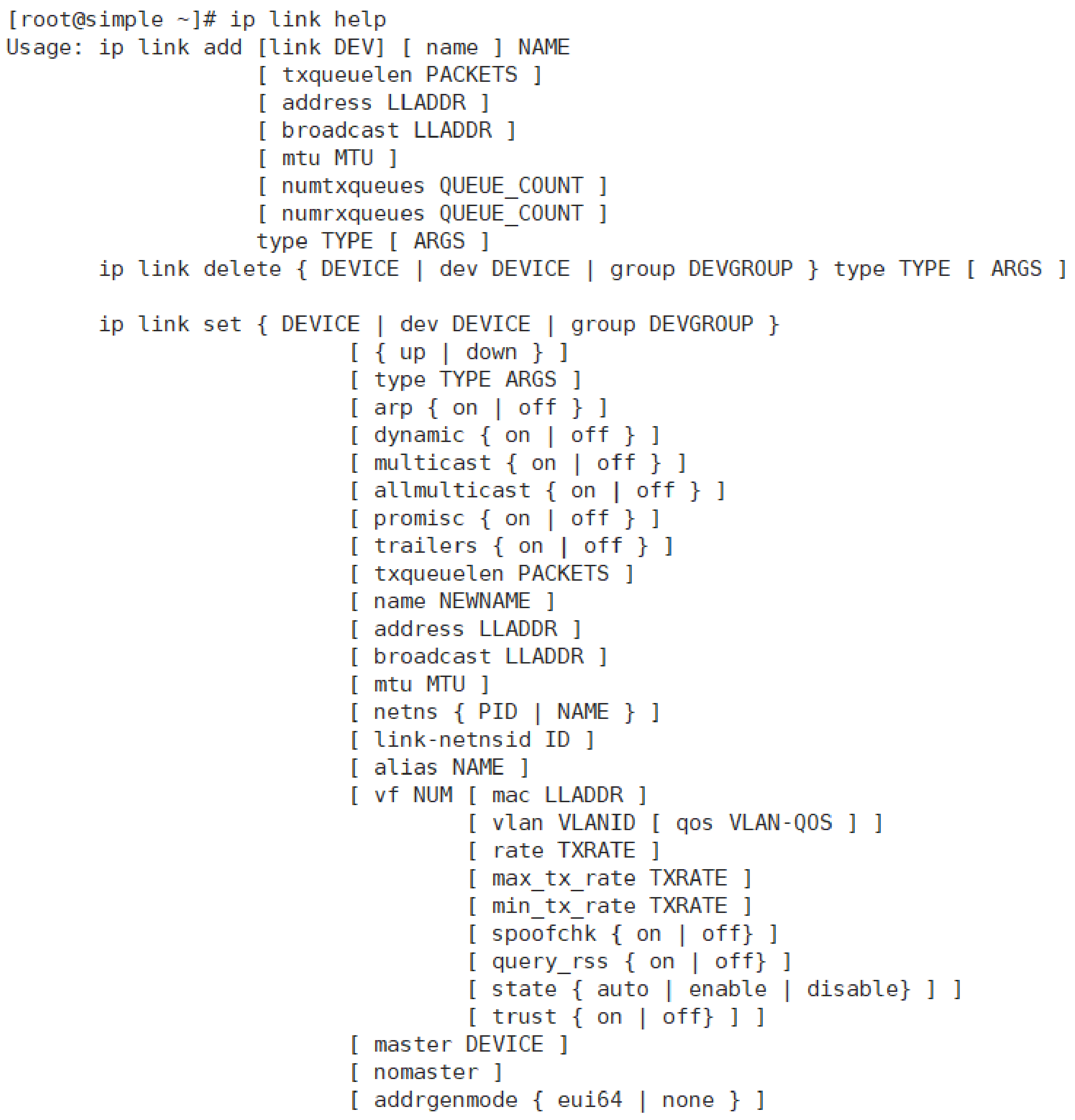


```
[root@simple ~]# yum -y install tcpdump bridge-utils
已加载插件：fastestmirror
Loading mirror speeds from cached hostfile
正在解决依赖关系
--> 正在检查事务
---> 软件包 bridge-utils.x86_64.0.1.5-9.el7 将被 安装
---> 软件包 tcpdump.x86_64.14.4.9.0-5.el7 将被 安装
--> 解决依赖关系完成

依赖关系解决

================================================================================================================
 Package                     架构                   版本                           源                      大小
================================================================================================================
正在安装:
 bridge-utils                x86_64                 1.5-9.el7                      simple                  32 k
 tcpdump                     x86_64                 14:4.9.0-5.el7                 simple                 415 k

事务概要
================================================================================================================
安装  2 软件包

总下载量：447 k
安装大小：1.1 M
Downloading packages:
----------------------------------------------------------------------------------------------------------------
总计                                                                              85 MB/s | 447 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  正在安装    : bridge-utils-1.5-9.el7.x86_64                                                              1/2
  正在安装    : 14:tcpdump-4.9.0-5.el7.x86_64                                                              2/2
  验证中      : 14:tcpdump-4.9.0-5.el7.x86_64                                                              1/2
  验证中      : bridge-utils-1.5-9.el7.x86_64                                                              2/2

已安装:
  bridge-utils.x86_64 0:1.5-9.el7                         tcpdump.x86_64 14:4.9.0-5.el7

完毕！
```

### 3.3 ip link 使用

(1)  查看 ip link 帮助

ip link help

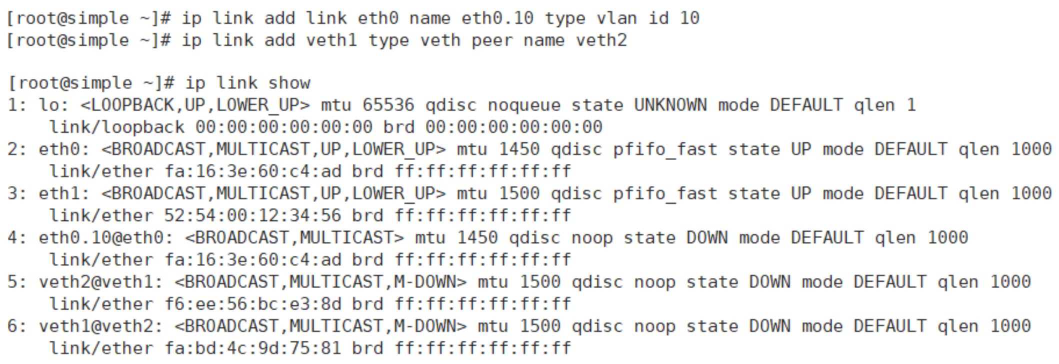


```
[root@simple ~]# ip link help
Usage: ip link add [link DEV] [ name ] NAME
                   [ txqueuelen PACKETS ]
                   [ address LLADDR ]
                   [ broadcast LLADDR ]
                   [ mtu MTU ]
                   [ numtxqueues QUEUE_COUNT ]
                   [ numrxqueues QUEUE_COUNT ]
                   type TYPE [ ARGS ]
       ip link delete { DEVICE | dev DEVICE | group DEVGROUP } type TYPE [ ARGS ]

       ip link set { DEVICE | dev DEVICE | group DEVGROUP }
                          [ { up | down } ]
                          [ type TYPE ARGS ]
                          [ arp { on | off } ]
                          [ dynamic { on | off } ]
                          [ multicast { on | off } ]
                          [ allmulticast { on | off } ]
                          [ promisc { on | off } ]
                          [ trailers { on | off } ]
                          [ txqueuelen PACKETS ]
                          [ name NEWNAME ]
                          [ address LLADDR ]
                          [ broadcast LLADDR ]
                          [ mtu MTU ]
                          [ netns { PID | NAME } ]
                          [ link-netnsid ID ]
                          [ alias NAME ]
                          [ vf NUM [ mac LLADDR ]
                                   [ vlan VLANID [ qos VLAN-QOS ] ]
                                   [ rate TXRATE ]
                                   [ max_tx_rate TXRATE ]
                                   [ min_tx_rate TXRATE ]
                                   [ spoofchk { on | off} ]
                                   [ query_rss { on | off} ]
                                   [ state { auto | enable | disable} ] ]
                                   [ trust { on | off} ] ]
                          [ master DEVICE ]
                          [ nomaster ]
                          [ addrgenmode { eui64 | none } ]
```

(2) 新建网络接口

ip link add link eth0 name eth0.10 type vlan id 10

ip link add veth1 type veth peer name veth2



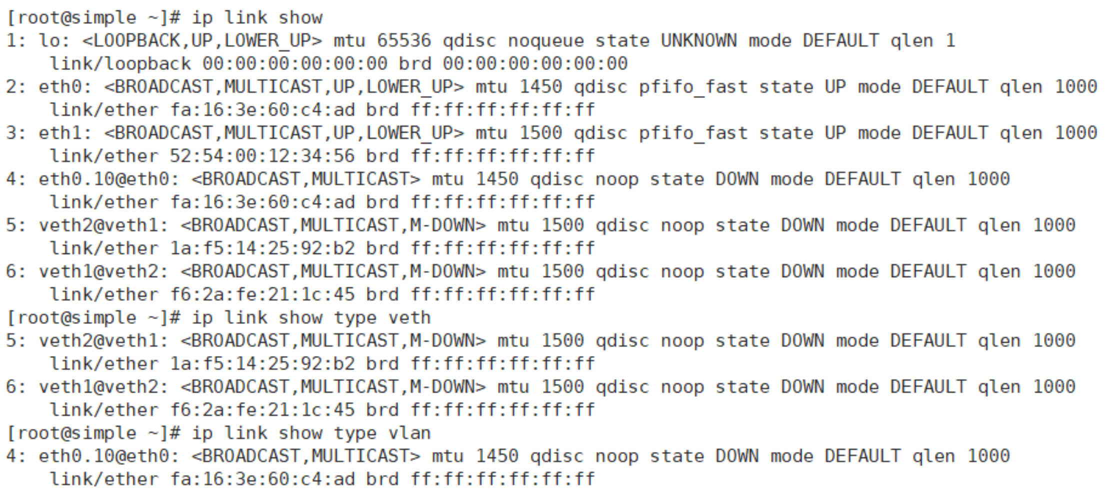
```
[root@simple ~]# ip link add link eth0 name eth0.10 type vlan id 10
[root@simple ~]# ip link add veth1 type veth peer name veth2

[root@simple ~]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether fa:16:3e:60:c4:ad brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff
4: eth0.10@eth0: <BROADCAST,MULTICAST> mtu 1450 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether fa:16:3e:60:c4:ad brd ff:ff:ff:ff:ff:ff
5: veth2@veth1: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether f6:ee:56:bc:e3:8d brd ff:ff:ff:ff:ff:ff
6: veth1@veth2: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether fa:bd:4c:9d:75:81 brd ff:ff:ff:ff:ff:ff
```

(3) 查看网络接口

ip link show

ip link show type veth

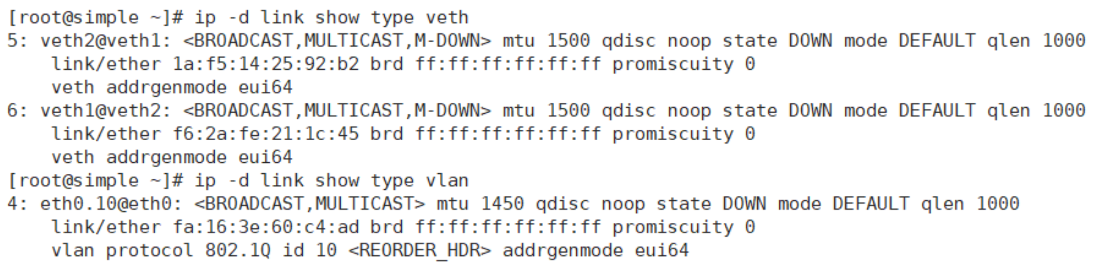
ip link show type vlan



```
[root@simple ~]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether fa:16:3e:60:c4:ad brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff
4: eth0.10@eth0: <BROADCAST,MULTICAST> mtu 1450 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether fa:16:3e:60:c4:ad brd ff:ff:ff:ff:ff:ff
5: veth2@veth1: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 1a:f5:14:25:92:b2 brd ff:ff:ff:ff:ff:ff
6: veth1@veth2: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether f6:2a:fe:21:1c:45 brd ff:ff:ff:ff:ff:ff
[root@simple ~]# ip link show type veth
5: veth2@veth1: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 1a:f5:14:25:92:b2 brd ff:ff:ff:ff:ff:ff
6: veth1@veth2: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether f6:2a:fe:21:1c:45 brd ff:ff:ff:ff:ff:ff
[root@simple ~]# ip link show type vlan
4: eth0.10@eth0: <BROADCAST,MULTICAST> mtu 1450 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether fa:16:3e:60:c4:ad brd ff:ff:ff:ff:ff:ff
```

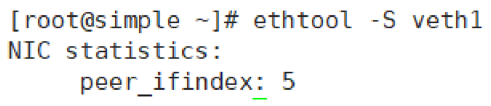
ip -d link show type veth

ip -d link show type vlan



```
[root@simple ~]# ip -d link show type veth
5: veth2@veth1: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 1a:f5:14:25:92:b2 brd ff:ff:ff:ff:ff:ff promiscuity 0
    veth addrgenmode eui64
6: veth1@veth2: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether f6:2a:fe:21:1c:45 brd ff:ff:ff:ff:ff:ff promiscuity 0
    veth addrgenmode eui64
[root@simple ~]# ip -d link show type vlan
4: eth0.10@eth0: <BROADCAST,MULTICAST> mtu 1450 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether fa:16:3e:60:c4:ad brd ff:ff:ff:ff:ff:ff promiscuity 0
    vlan protocol 802.1Q id 10 <REORDER_HDR> addrgenmode eui64
```

ethtool -S veth1



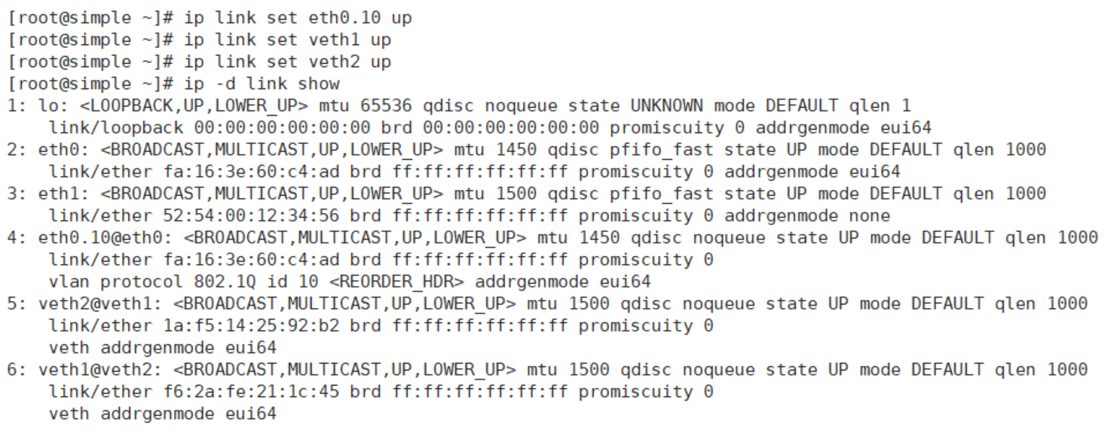
```
[root@simple ~]# ethtool -S veth1
NIC statistics:
     peer_ifindex: 5
```

(4) 使接口 UP

ip link set eth0.10 up

ip link set veth1 up

ip link set veth2 up

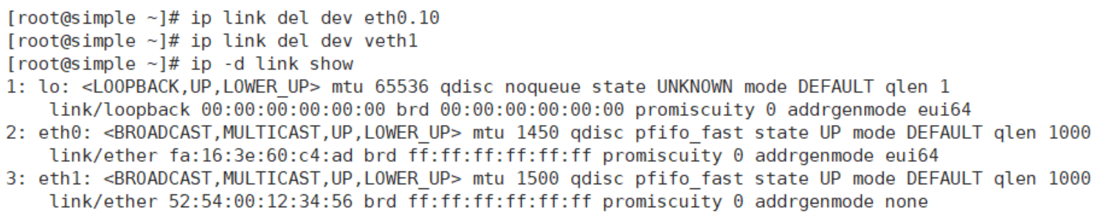


```
[root@simple ~]# ip link set eth0.10 up
[root@simple ~]# ip link set veth1 up
[root@simple ~]# ip link set veth2 up
[root@simple ~]# ip -d link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 promiscuity 0 addrgenmode eui64
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether fa:16:3e:60:c4:ad brd ff:ff:ff:ff:ff:ff promiscuity 0 addrgenmode eui64
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff promiscuity 0 addrgenmode none
4: eth0.10@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UP mode DEFAULT qlen 1000
    link/ether fa:16:3e:60:c4:ad brd ff:ff:ff:ff:ff:ff promiscuity 0
    vlan protocol 802.1Q id 10 <REORDER_HDR> addrgenmode eui64
5: veth2@veth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT qlen 1000
    link/ether 1a:f5:14:25:92:b2 brd ff:ff:ff:ff:ff:ff promiscuity 0
    veth addrgenmode eui64
6: veth1@veth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT qlen 1000
    link/ether f6:2a:fe:21:1c:45 brd ff:ff:ff:ff:ff:ff promiscuity 0
    veth addrgenmode eui64
```

(5) 删除接口

ip link del dev eth0.10

ip link del dev veth1

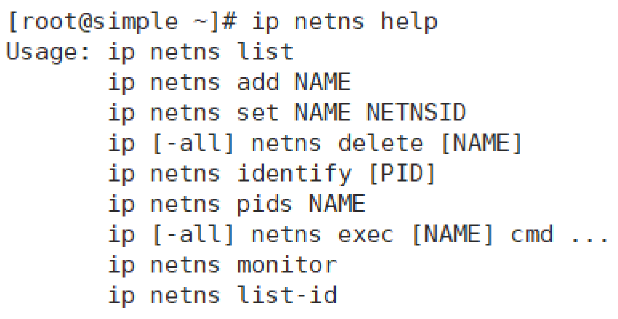


```
[root@simple ~]# ip link del dev eth0.10
[root@simple ~]# ip link del dev veth1
[root@simple ~]# ip -d link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 promiscuity 0 addrgenmode eui64
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether fa:16:3e:60:c4:ad brd ff:ff:ff:ff:ff:ff promiscuity 0 addrgenmode eui64
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff promiscuity 0 addrgenmode none
```

### 3.4 ip netns 使用

(1) 查看 ip netns 帮助

ip netns help



```
[root@simple ~]# ip netns help
Usage: ip netns list
       ip netns add NAME
       ip netns set NAME NETNSID
       ip [-all] netns delete [NAME]
       ip netns identify [PID]
       ip netns pids NAME
       ip [-all] netns exec [NAME] cmd ...
       ip netns monitor
       ip netns list-id
```

(2) 创建一个名为 test 的 namespace

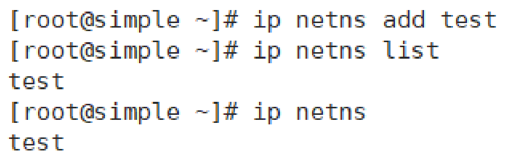
ip netns add test

(3) 查看所有 namespace

以下两条命令都可以查看所有 namespace。

ip netns list

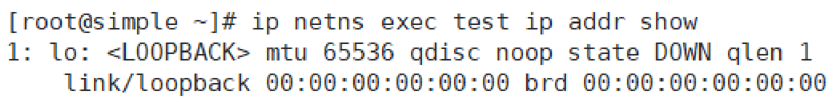
ip netns



```
[root@simple ~]# ip netns add test
[root@simple ~]# ip netns list
test
[root@simple ~]# ip netns
test
```

(4) 查看名为 test 的 namespace

ip netns exec test ip addr show



```
[root@simple ~]# ip netns exec test ip addr show
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

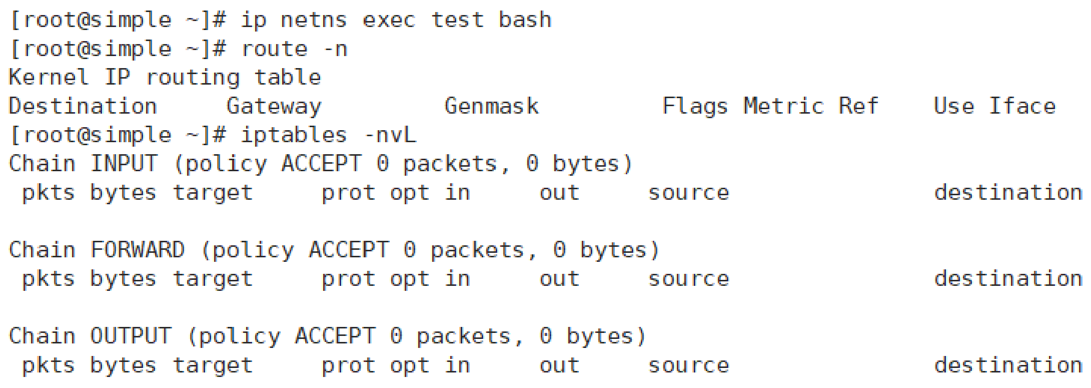
(5) 进入名为 test 的 namespace,执行网络命令

使用 ip netns 的 bash 参数可以进入 namespace。

ip netns exec test bash

route -n

iptables -nvL



```
[root@simple ~]# ip netns exec test bash
[root@simple ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

(6) 退出 namespace

使用命令 exit 可以退出当前 namespace。

exit

(7)给 test 添加接口 tap1

首先新建接口 tap1。

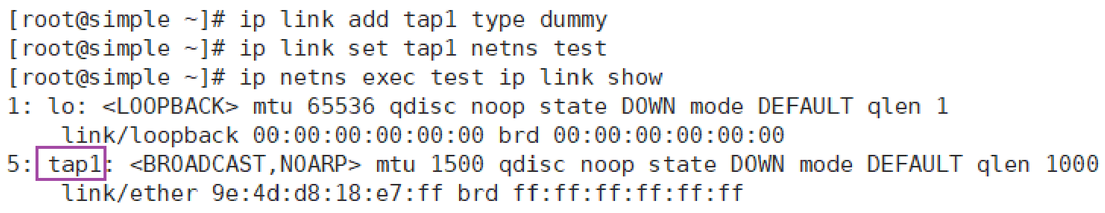
ip link add tap1 type dummy

然后给 test 添加接口 tap1。

ip link set tap1 netns test

查看 test 启用的接口，能够发现 tap1。

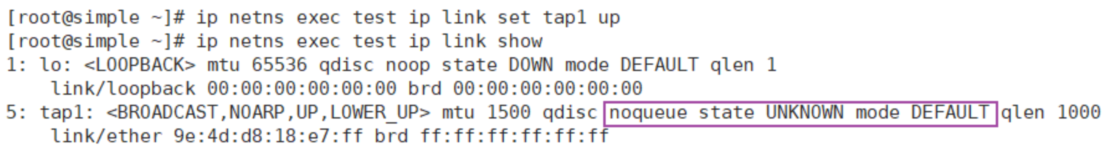
ip netns exec test ip link show



```
[root@simple ~]# ip link add tap1 type dummy
[root@simple ~]# ip link set tap1 netns test
[root@simple ~]# ip netns exec test ip link show
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
5: tap1: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 9e:4d:d8:18:e7:ff brd ff:ff:ff:ff:ff:ff
```

(8) 启用 tap1 虚拟接口

ip netns exec test ip link set tap1 up



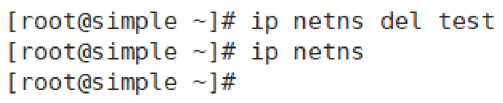
```
[root@simple ~]# ip netns exec test ip link set tap1 up
[root@simple ~]# ip netns exec test ip link show
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
5: tap1: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1000
    link/ether 9e:4d:d8:18:e7:ff brd ff:ff:ff:ff:ff:ff
```

(9) 给 tap1 虚拟接口配置 IP

ip netns exec test ip addr add dev tap1 192.168.0.1/24

(10) 删除 test namespace

ip netns del test

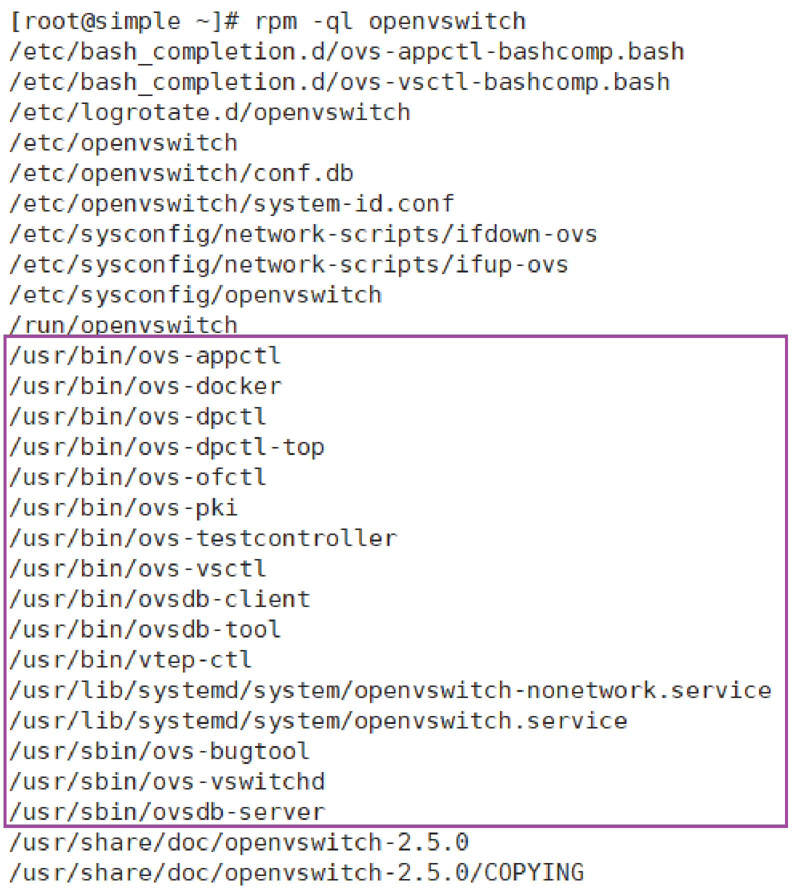


```
[root@simple ~]# ip netns del test
[root@simple ~]# ip netns
[root@simple ~]#
```

**3.5 openvswitch 使用**

(1) 查看 openvswitch 安装的命令工具

rpm -ql openvswitch



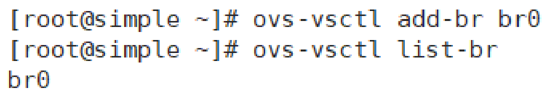
```
[root@simple ~]# rpm -ql openvswitch
/etc/bash_completion.d/ovs-appctl-bashcomp.bash
/etc/bash_completion.d/ovs-vsctl-bashcomp.bash
/etc/logrotate.d/openvswitch
/etc/openvswitch
/etc/openvswitch/conf.db
/etc/openvswitch/system-id.conf
/etc/sysconfig/network-scripts/ifdown-ovs
/etc/sysconfig/network-scripts/ifup-ovs
/etc/sysconfig/openvswitch
/run/openvswitch
/usr/bin/ovs-appctl
/usr/bin/ovs-docker
/usr/bin/ovs-dpctl
/usr/bin/ovs-dpctl-top
/usr/bin/ovs-ofctl
/usr/bin/ovs-pki
/usr/bin/ovs-testcontroller
/usr/bin/ovs-vsctl
/usr/bin/ovsdb-client
/usr/bin/ovsdb-tool
/usr/bin/vtep-ctl
/usr/lib/systemd/system/openvswitch-nonetwork.service
/usr/lib/systemd/system/openvswitch.service
/usr/sbin/ovs-bugtool
/usr/sbin/ovs-vswitchd
/usr/sbin/ovsdb-server
/usr/share/doc/openvswitch-2.5.0
/usr/share/doc/openvswitch-2.5.0/COPYING
```

(2)添加网桥 br0

ovs-vsctl add-br br0

(3)列出 openvswitch 中所有的网桥

ovs-vsctl list-br



```
[root@simple ~]# ovs-vsctl add-br br0
[root@simple ~]# ovs-vsctl list-br
br0
```

(4)判断网桥是否存在

ovs-vsctl br-exists br0

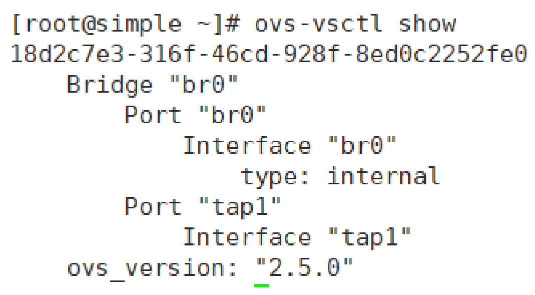
(5)将网卡添加到网桥 br0

ip link add tap1 type dummy

ovs-vsctl add-port br0 tap1

(6)查看 openvswitch 的网络状态

ovs-vsctl show



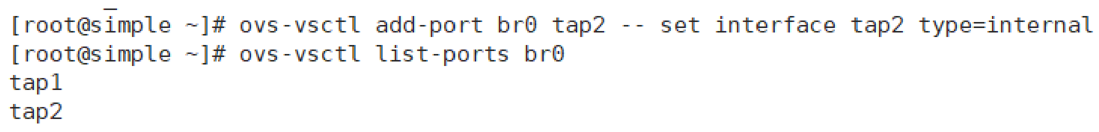
```
[root@simple ~]# ovs-vsctl show
18d2c7e3-316f-46cd-928f-8ed0c2252fe0
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
        Port "tap1"
            Interface "tap1"
    ovs_version: "2.5.0"
```

(7)在网桥 br0 中新建 openvswitch 网口

ovs-vsctl add-port br0 tap2 -- set interface tap2 type=internal

(8)列出网桥 br0 中所有端口

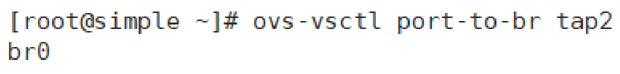
ovs-vsctl list-ports br0



```
[root@simple ~]# ovs-vsctl add-port br0 tap2 -- set interface tap2 type=internal
[root@simple ~]# ovs-vsctl list-ports br0
tap1
tap2
```

(9)列出所有连接到网卡 tap2 的网桥

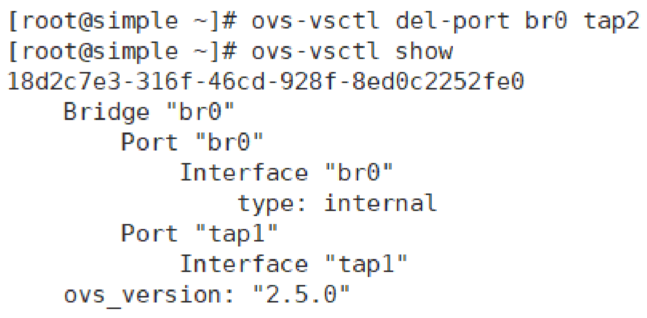
ovs-vsctl port-to-br tap2



```
[root@simple ~]# ovs-vsctl port-to-br tap2
br0
```

(10)删除网桥 br0 上的网口 tap2

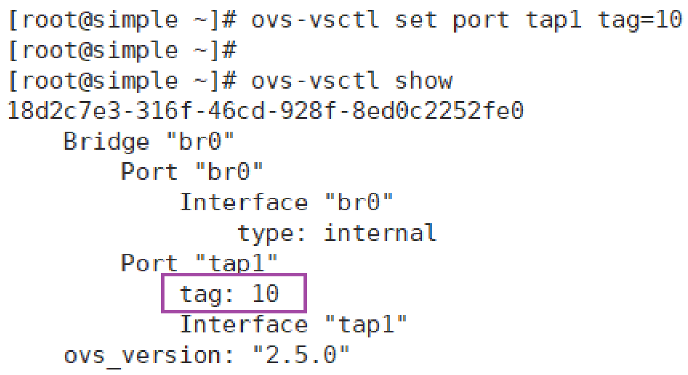
ovs-vsctl del-port br0 tap2



```
[root@simple ~]# ovs-vsctl del-port br0 tap2
[root@simple ~]# ovs-vsctl show
18d2c7e3-316f-46cd-928f-8ed0c2252fe0
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
        Port "tap1"
            Interface "tap1"
    ovs_version: "2.5.0"
```

(11)设置网口 tap1 的 vlan tag 为 10

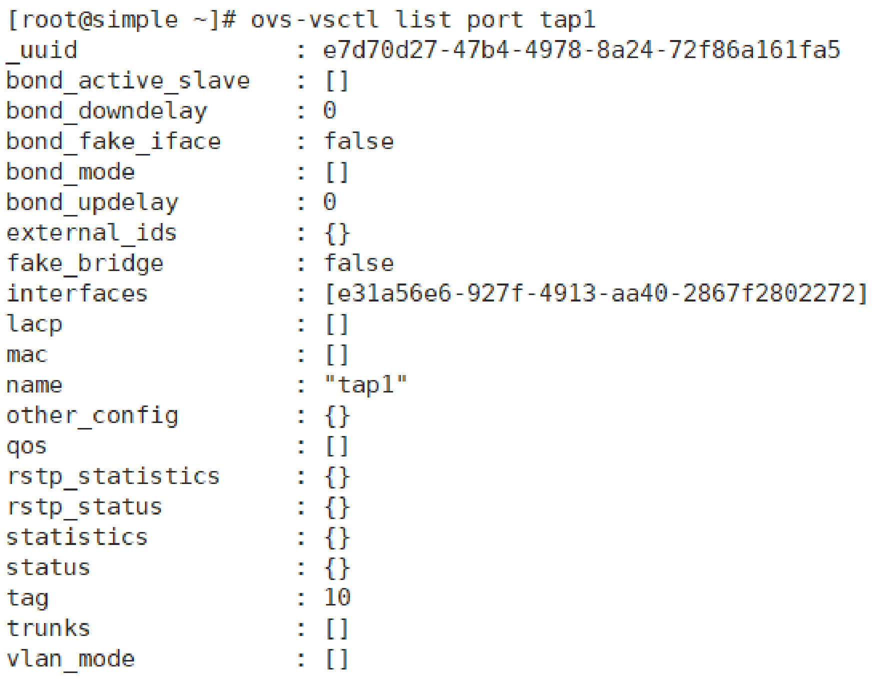
ovs-vsctl set port tap1 tag=10



```
[root@simple ~]# ovs-vsctl set port tap1 tag=10
[root@simple ~]#
[root@simple ~]# ovs-vsctl show
18d2c7e3-316f-46cd-928f-8ed0c2252fe0
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
        Port "tap1"
            tag: 10
            Interface "tap1"
    ovs_version: "2.5.0"
```

(12)查看网口 tap1 的属性

ovs-vsctl list port tap1



```
[root@simple ~]# ovs-vsctl list port tap1
_uuid               : e7d70d27-47b4-4978-8a24-72f86a161fa5
bond_active_slave   : []
bond_downdelay      : 0
bond_fake_iface     : false
bond_mode           : []
bond_updelay        : 0
external_ids        : {}
fake_bridge         : false
interfaces          : [e31a56e6-927f-4913-aa40-2867f2802272]
lacp                : []
mac                 : []
name                : "tap1"
other_config        : {}
qos                 : []
rstp_statistics     : {}
rstp_status         : {}
statistics          : {}
status              : {}
tag                 : 10
trunks              : []
vlan_mode           : []
```

(13)从网桥 br0 删除网口 tap1，并从系统删除虚拟网口 tap1

首先从网桥 br0 删除网口 tap1。

ovs-vsctl del-port br0 tap1

然后从系统删除虚拟网口 tap1。

ip link del dev tap1

```
[root@simple ~]# ovs-vsctl del-port br0 tap1
[root@simple ~]# ip link del dev tap1
[root@simple ~]# ovs-vsctl show
18d2c7e3-316f-46cd-928f-8ed0c2252fe0
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "2.5.0"
```

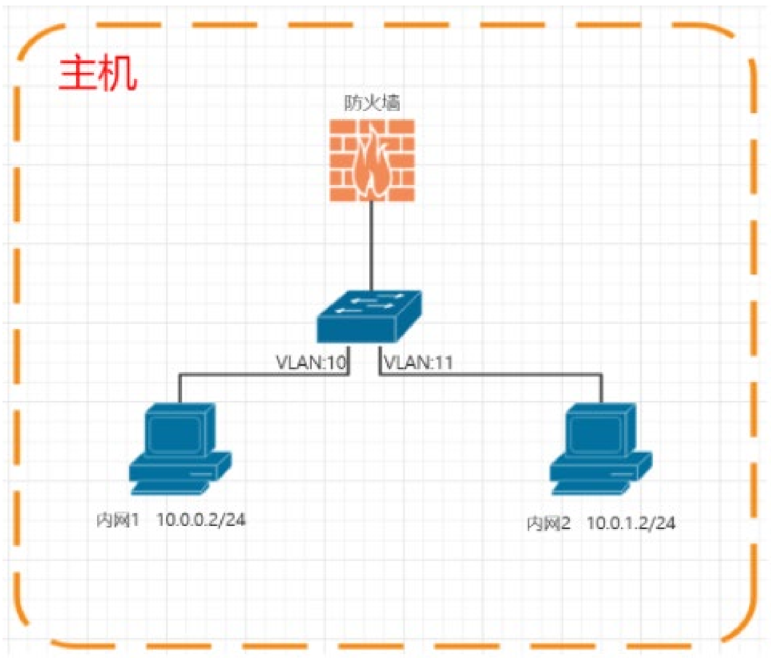
(14)删除网桥 br0

ovs-vsctl del-br br0

```
[root@simple ~]# ovs-vsctl del-br br0
[root@simple ~]# ovs-vsctl show
18d2c7e3-316f-46cd-928f-8ed0c2252fe0
    ovs_version: "2.5.0"
```

## 任务四

### 4.1 实验网络拓扑



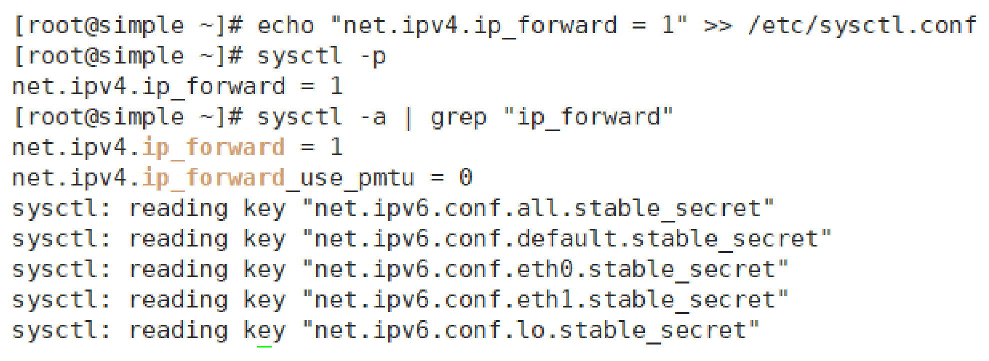
### 4.2 环境搭建

1、开启主机路由转发功能

echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf

sysctl -p

sysctl -a | grep "ip_forward"



```
[root@simple ~]# echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
[root@simple ~]# sysctl -p
net.ipv4.ip_forward = 1
[root@simple ~]# sysctl -a | grep "ip_forward"
net.ipv4.ip_forward = 1
net.ipv4.ip_forward_use_pmtu = 0
sysctl: reading key "net.ipv6.conf.all.stable_secret"
sysctl: reading key "net.ipv6.conf.default.stable_secret"
sysctl: reading key "net.ipv6.conf.eth0.stable_secret"
sysctl: reading key "net.ipv6.conf.eth1.stable_secret"
sysctl: reading key "net.ipv6.conf.lo.stable_secret"
```

2、创建 tag 为 10 的内网 1

首先需要添加网桥 br0。

ovs-vsctl add-br br0

然后创建 tag 为 10 的内网 1，名为 tap1。

ovs-vsctl add-port br0 tap1 tag=10 -- set interface tap1 type=internal

在查看信息后，创建一个名为 ns-tap1 的 namespace。

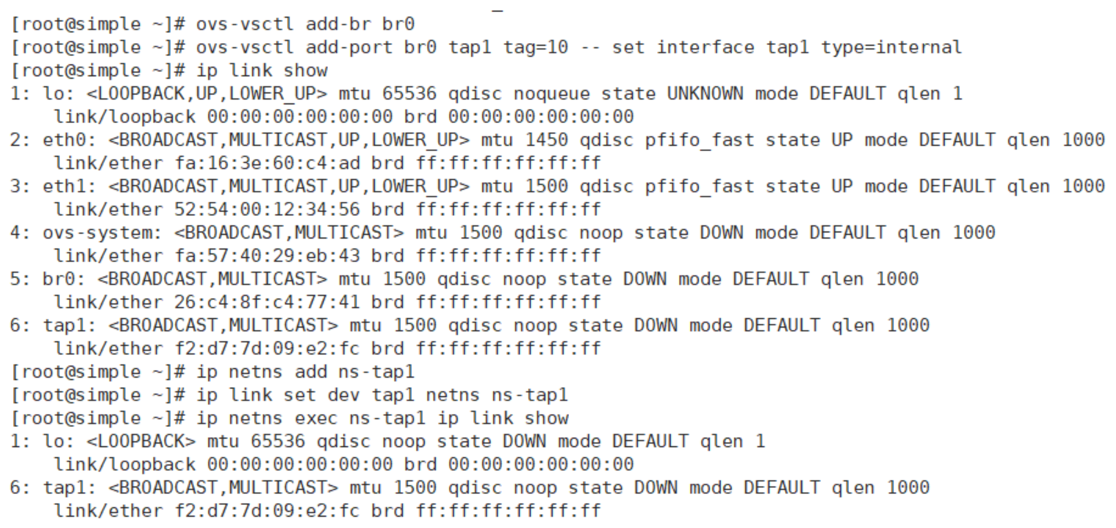
ip link show

ip netns add ns-tap1

然后给 ns-tap1 添加接口 tap1，再查看 ns-tap1 启用的接口。

ip link set dev tap1 netns ns-tap1

ip netns exec ns-tap1 ip link show

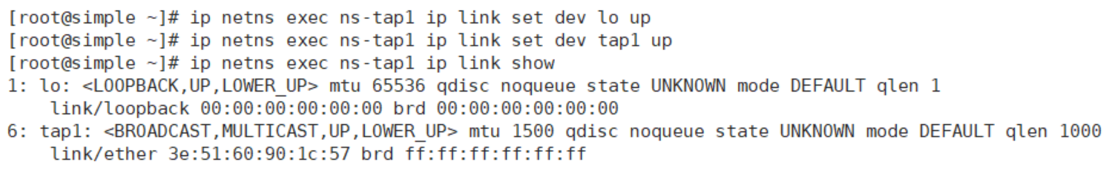


```
[root@simple ~]# ovs-vsctl add-br br0
[root@simple ~]# ovs-vsctl add-port br0 tap1 tag=10 -- set interface tap1 type=internal
[root@simple ~]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether fa:16:3e:60:c4:ad brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff
4: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether fa:57:40:29:eb:43 brd ff:ff:ff:ff:ff:ff
5: br0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 26:c4:8f:c4:77:41 brd ff:ff:ff:ff:ff:ff
6: tap1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether f2:d7:7d:09:e2:fc brd ff:ff:ff:ff:ff:ff
[root@simple ~]# ip netns add ns-tap1
[root@simple ~]# ip link set dev tap1 netns ns-tap1
[root@simple ~]# ip netns exec ns-tap1 ip link show
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
6: tap1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether f2:d7:7d:09:e2:fc brd ff:ff:ff:ff:ff:ff
```

设置命名空间 ns-tap1 的属性，并且启用 tap1 接口。

ip netns exec ns-tap1 ip link set dev lo up

ip netns exec ns-tap1 ip link set dev tap1 up



```
[root@simple ~]# ip netns exec ns-tap1 ip link set dev lo up
[root@simple ~]# ip netns exec ns-tap1 ip link set dev tap1 up
[root@simple ~]# ip netns exec ns-tap1 ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
6: tap1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1000
    link/ether 3e:51:60:90:1c:57 brd ff:ff:ff:ff:ff:ff
```

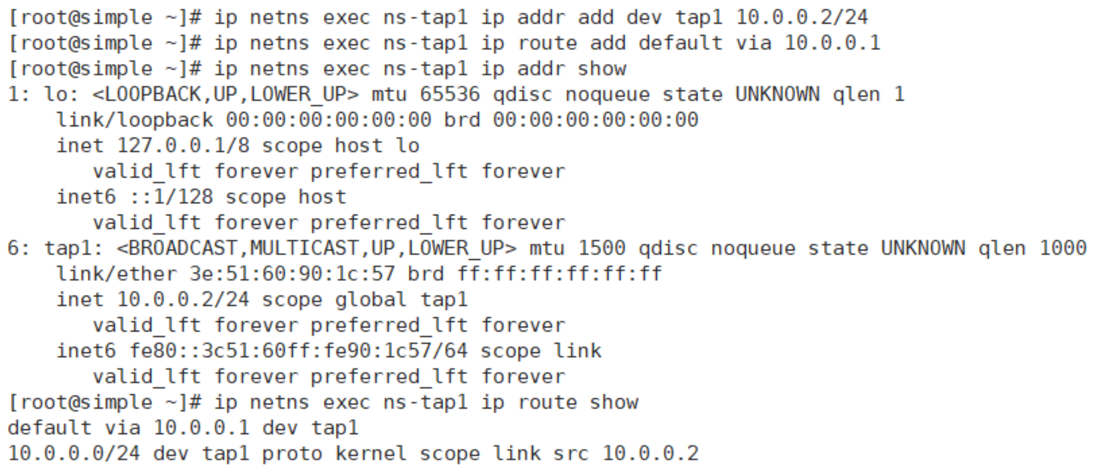
给 tap1 接口配置 IP。

ip netns exec ns-tap1 ip addr add dev tap1 10.0.0.2/24

ip netns exec ns-tap1 ip route add default via 10.0.0.1

ip netns exec ns-tap1 ip addr show

ip netns exec ns-tap1 ip route show



```
[root@simple ~]# ip netns exec ns-tap1 ip addr add dev tap1 10.0.0.2/24
[root@simple ~]# ip netns exec ns-tap1 ip route add default via 10.0.0.1
[root@simple ~]# ip netns exec ns-tap1 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
6: tap1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN qlen 1000
    link/ether 3e:51:60:90:1c:57 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.2/24 scope global tap1
       valid_lft forever preferred_lft forever
    inet6 fe80::3c51:60ff:fe90:1c57/64 scope link
       valid_lft forever preferred_lft forever
[root@simple ~]# ip netns exec ns-tap1 ip route show
default via 10.0.0.1 dev tap1
10.0.0.0/24 dev tap1 proto kernel scope link src 10.0.0.2
```

3、创建 tag 为 11 的内网 2，方法同上

将命名空间命名为 ns-tap2，内网 2 命名为 tap2。为内网 2 和路由分配的 IP 地址需要与内网 1 区分开来。

ovs-vsctl add-port br0 tap2 tag=11 -- set interface tap2 type=internal

ip link show

ip netns add ns-tap2

ip link set dev tap2 netns ns-tap2

ip netns exec ns-tap2 ip link show

ip netns exec ns-tap2 ip link set dev lo up

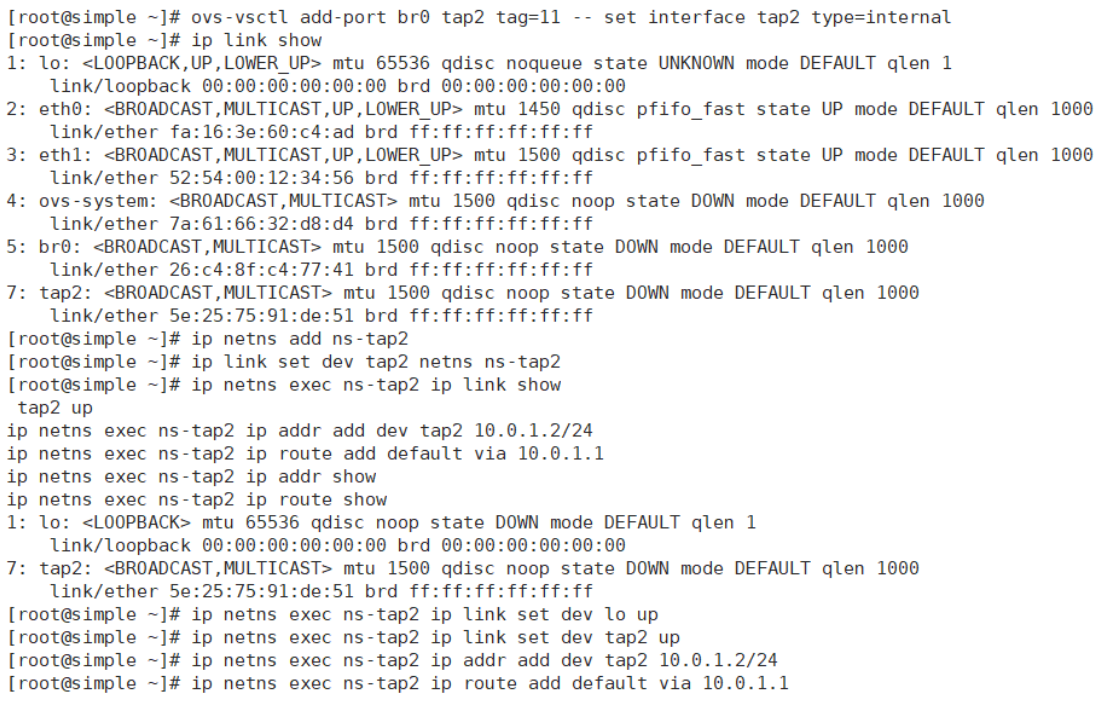
ip netns exec ns-tap2 ip link set dev tap2 up

ip netns exec ns-tap2 ip addr add dev tap2 10.0.1.2/24

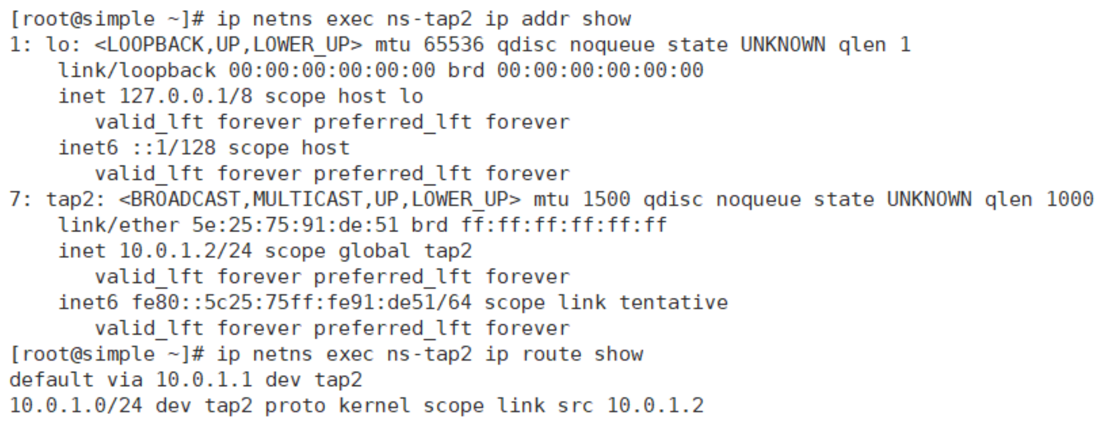
ip netns exec ns-tap2 ip route add default via 10.0.1.1

ip netns exec ns-tap2 ip addr show

ip netns exec ns-tap2 ip route show



```
[root@simple ~]# ovs-vsctl add-port br0 tap2 tag=11 -- set interface tap2 type=internal
[root@simple ~]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether fa:16:3e:60:c4:ad brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff
4: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 7a:61:66:32:d8:d4 brd ff:ff:ff:ff:ff:ff
5: br0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 26:c4:8f:c4:77:41 brd ff:ff:ff:ff:ff:ff
7: tap2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 5e:25:75:91:de:51 brd ff:ff:ff:ff:ff:ff
[root@simple ~]# ip netns add ns-tap2
[root@simple ~]# ip link set dev tap2 netns ns-tap2
[root@simple ~]# ip netns exec ns-tap2 ip link show
 tap2 up
ip netns exec ns-tap2 ip addr add dev tap2 10.0.1.2/24
ip netns exec ns-tap2 ip route add default via 10.0.1.1
ip netns exec ns-tap2 ip addr show
ip netns exec ns-tap2 ip route show
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
7: tap2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 5e:25:75:91:de:51 brd ff:ff:ff:ff:ff:ff
[root@simple ~]# ip netns exec ns-tap2 ip link set dev lo up
[root@simple ~]# ip netns exec ns-tap2 ip link set dev tap2 up
[root@simple ~]# ip netns exec ns-tap2 ip addr add dev tap2 10.0.1.2/24
[root@simple ~]# ip netns exec ns-tap2 ip route add default via 10.0.1.1
```



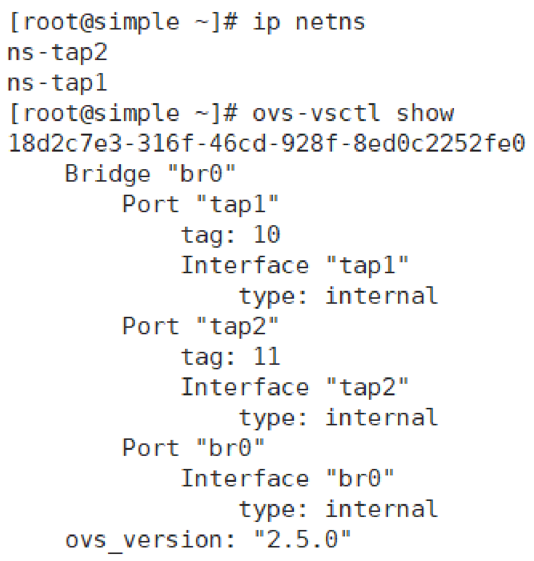
```
[root@simple ~]# ip netns exec ns-tap2 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
7: tap2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN qlen 1000
    link/ether 5e:25:75:91:de:51 brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.2/24 scope global tap2
       valid_lft forever preferred_lft forever
    inet6 fe80::5c25:75ff:fe91:de51/64 scope link tentative
       valid_lft forever preferred_lft forever
[root@simple ~]# ip netns exec ns-tap2 ip route show
default via 10.0.1.1 dev tap2
10.0.1.0/24 dev tap2 proto kernel scope link src 10.0.1.2
```

4、查看网络内网 1 与 内网 2 的连通性

查看当前内网信息。

ip netns

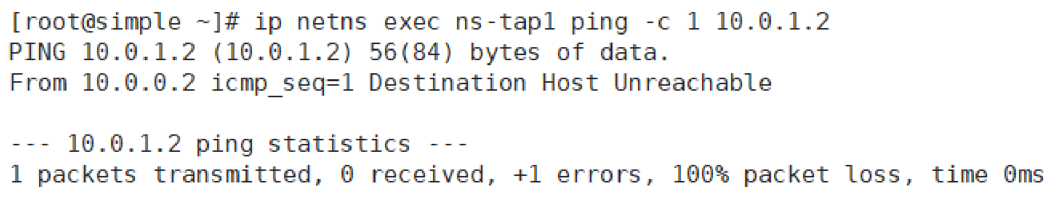
ovs-vsctl show



```
[root@simple ~]# ip netns
ns-tap2
ns-tap1
[root@simple ~]# ovs-vsctl show
18d2c7e3-316f-46cd-928f-8ed0c2252fe0
    Bridge "br0"
        Port "tap1"
            tag: 10
            Interface "tap1"
                type: internal
        Port "tap2"
            tag: 11
            Interface "tap2"
                type: internal
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "2.5.0"
```

网络内网 1 和内网 2 是不连通的。

ip netns exec ns-tap1 ping -c 1 10.0.1.2



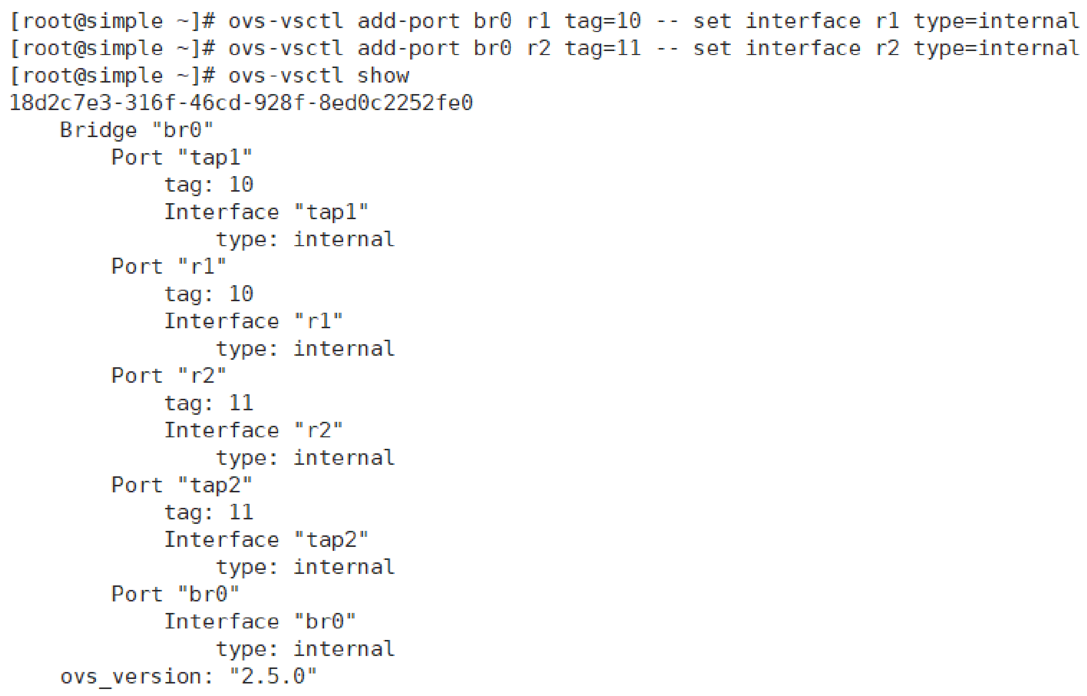
```
[root@simple ~]# ip netns exec ns-tap1 ping -c 1 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
From 10.0.0.2 icmp_seq=1 Destination Host Unreachable

--- 10.0.1.2 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

5、netns 模拟路由器实现内网 1 与内网 2 互通

首先在网桥 br0 中新建端口 r1 和 r2。

ovs-vsctl add-port br0 r1 tag=10 -- set interface r1 type=internal

ovs-vsctl add-port br0 r2 tag=11 -- set interface r2 type=internal



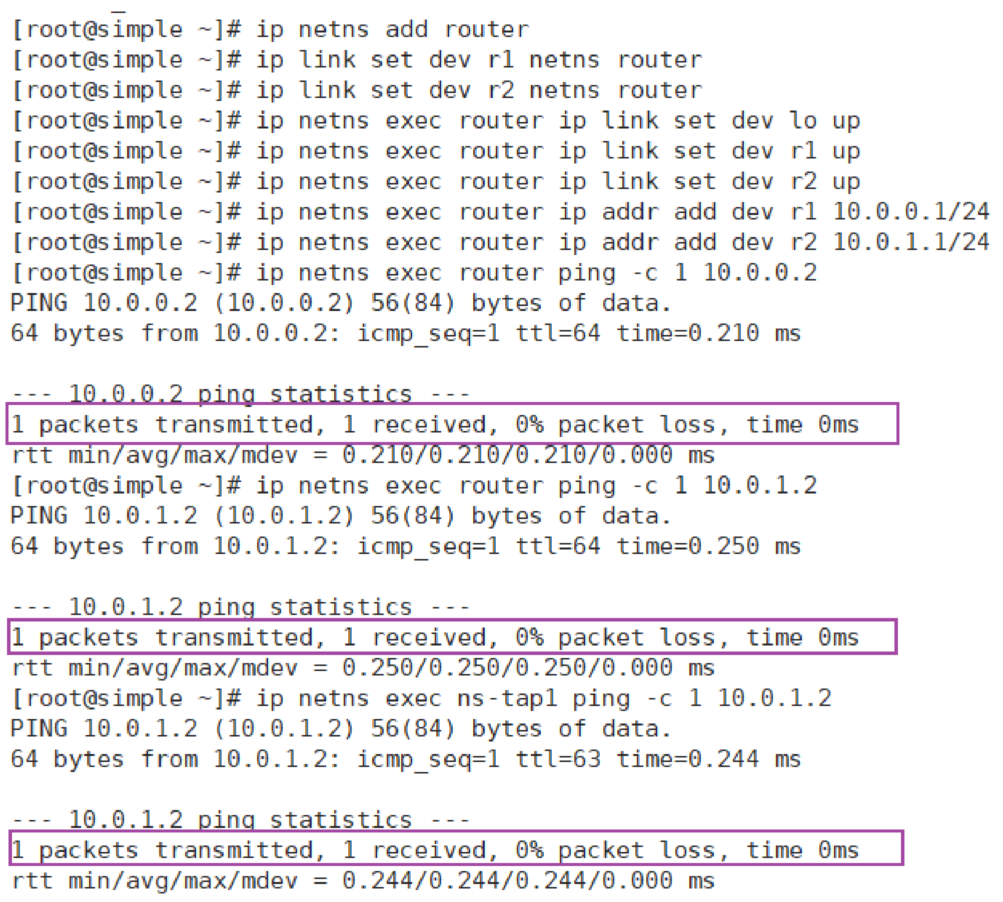
```
[root@simple ~]# ovs-vsctl add-port br0 r1 tag=10 -- set interface r1 type=internal
[root@simple ~]# ovs-vsctl add-port br0 r2 tag=11 -- set interface r2 type=internal
[root@simple ~]# ovs-vsctl show
18d2c7e3-316f-46cd-928f-8ed0c2252fe0
    Bridge "br0"
        Port "tap1"
            tag: 10
            Interface "tap1"
                type: internal
        Port "r1"
            tag: 10
            Interface "r1"
                type: internal
        Port "r2"
            tag: 11
            Interface "r2"
                type: internal
        Port "tap2"
            tag: 11
            Interface "tap2"
                type: internal
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "2.5.0"
```

接着设置路由信息。

```
ip netns add router
ip link set dev r1 netns router
ip link set dev r2 netns router
ip netns exec router ip link set dev lo up
ip netns exec router ip link set dev r1 up
ip netns exec router ip link set dev r2 up
ip netns exec router ip addr add dev r1 10.0.0.1/24
ip netns exec router ip addr add dev r2 10.0.1.1/24
```

最后测试内网 1 和内网 2 是否能够连通，可以发现此时内网 1 和内网 2 是互通的。路由器与两个内网之间是互通的，内网 1 和内网 2 之间也是互通的。

```
ip netns exec router ping -c 1 10.0.0.2
ip netns exec router ping -c 1 10.0.1.2
ip netns exec ns-tap1 ping -c 1 10.0.1.2
```



```
[root@simple ~]# ip netns add router
[root@simple ~]# ip link set dev r1 netns router
[root@simple ~]# ip link set dev r2 netns router
[root@simple ~]# ip netns exec router ip link set dev lo up
[root@simple ~]# ip netns exec router ip link set dev r1 up
[root@simple ~]# ip netns exec router ip link set dev r2 up
[root@simple ~]# ip netns exec router ip addr add dev r1 10.0.0.1/24
[root@simple ~]# ip netns exec router ip addr add dev r2 10.0.1.1/24
[root@simple ~]# ip netns exec router ping -c 1 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.210 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.210/0.210/0.210/0.000 ms
[root@simple ~]# ip netns exec router ping -c 1 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=0.250 ms

--- 10.0.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.250/0.250/0.250/0.000 ms
[root@simple ~]# ip netns exec ns-tap1 ping -c 1 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=63 time=0.244 ms

--- 10.0.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.244/0.244/0.244/0.000 ms
```

## 任务五

### 5.1 iptables nat 基础知识

nat 表需要的三个链:

PREROUTING:在数据包到达防火墙时进行路由判断之前的规则，作用是是否改变目的地址或者目的端口;

POSTROUTING:在数据包离开防火墙时进行路由判断，是否要改变源地址、源端口等;

INPUT:改变访问目的为主机的数据包源地址;

OUTPUT:改变主机发出去的数据包目的地址。

动作选项：

REDIRECT:  将数据包重定向到其它或其它主机的某个端口;

SNAT:  源地址转换，改变数据包的源地址;

DNAT:  目的地址转换，改变数据包的目的地址;

MASQUERADE: ip 智能伪装。

注意点:

PRERROUTING: DNAT、REDIRECT （路由之前）只支持-i，不支持-o。在作出路由之前，对目的地址进行修改;

POSTROUTING: SNAT、MASQUERADE （路由之后）只支持-o，不支持-i。在作出路由之后，对源地址进行修改;

OUTPUT: DNAT、REDIRECT （本机）DNAT 和 REDIRECT 规则用来处理来自 NAT 主机的出站数据包;

INPUT: SNAT （本机）SNAT 规则用来修改目的为本机的源地址。

**5.2 nat 常用配置**

(1) 将源转换成路由器 router 的 r2 接口地址

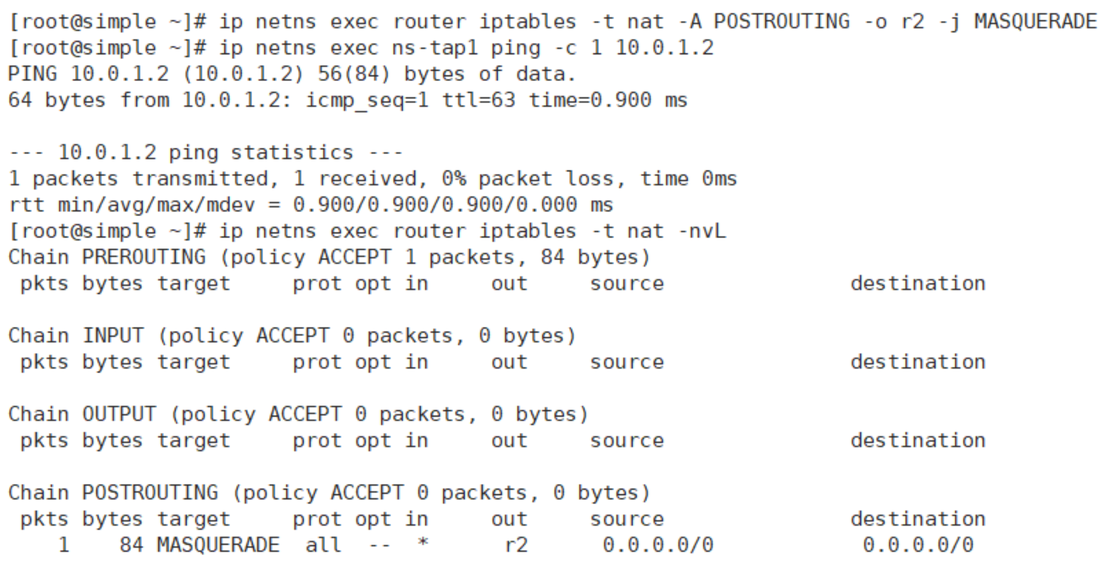
首先将源转换成路由器 router 的 r2 接口地址。

ip netns exec router iptables -t nat -A POSTROUTING -o r2 -j MASQUERADE

然后使用 ping 查看是否连通，再查看路由规则信息。

ip netns exec ns-tap1 ping -c 1 10.0.1.2

ip netns exec router iptables -t nat -nvL



```
[root@simple ~]# ip netns exec router iptables -t nat -A POSTROUTING -o r2 -j MASQUERADE
[root@simple ~]# ip netns exec ns-tap1 ping -c 1 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=63 time=0.900 ms

--- 10.0.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.900/0.900/0.900/0.000 ms
[root@simple ~]# ip netns exec router iptables -t nat -nvL
Chain PREROUTING (policy ACCEPT 1 packets, 84 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1    84 MASQUERADE  all  --  *      r2      0.0.0.0/0            0.0.0.0/0
```

(2) 打开两个终端利用 tcpdump 抓包分析

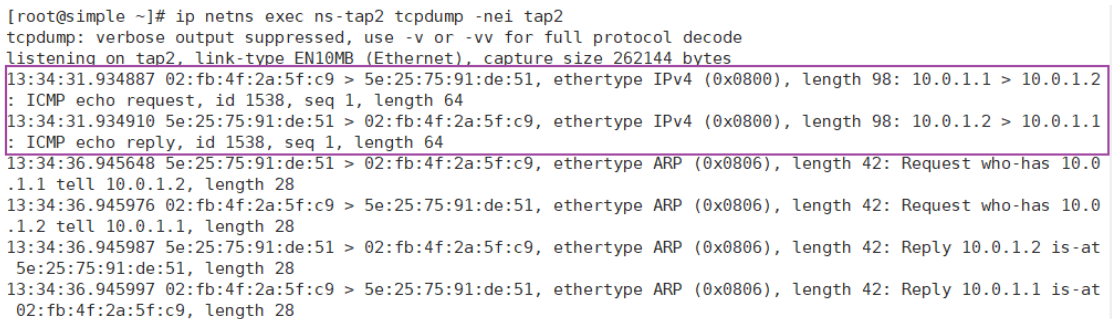
在一个终端使用如下命令进行流量监听。

ip netns exec ns-tap2 tcpdump -nei tap2

在另一个终端使用 ping 命令。

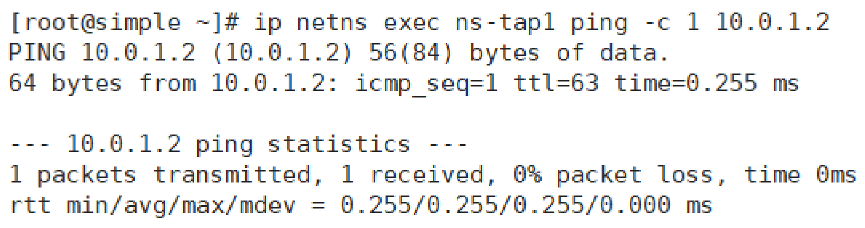
ip netns exec ns-tap1 ping -c 1 10.0.1.2

可以看到负责监听的终端收到了 ping 命令发送的 ICMP 数据包。



```
[root@simple ~]# ip netns exec ns-tap2 tcpdump -nei tap2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tap2, link-type EN10MB (Ethernet), capture size 262144 bytes
13:34:31.934887 02:fb:4f:2a:5f:c9 > 5e:25:75:91:de:51, ethertype IPv4 (0x0800), length 98: 10.0.1.1 > 10.0.1.2
: ICMP echo request, id 1538, seq 1, length 64
13:34:31.934910 5e:25:75:91:de:51 > 02:fb:4f:2a:5f:c9, ethertype IPv4 (0x0800), length 98: 10.0.1.2 > 10.0.1.1
: ICMP echo reply, id 1538, seq 1, length 64
13:34:36.945648 5e:25:75:91:de:51 > 02:fb:4f:2a:5f:c9, ethertype ARP (0x0806), length 42: Request who-has 10.0
.1.1 tell 10.0.1.2, length 28
13:34:36.945976 02:fb:4f:2a:5f:c9 > 5e:25:75:91:de:51, ethertype ARP (0x0806), length 42: Request who-has 10.0
.1.2 tell 10.0.1.1, length 28
13:34:36.945987 5e:25:75:91:de:51 > 02:fb:4f:2a:5f:c9, ethertype ARP (0x0806), length 42: Reply 10.0.1.2 is-at
 5e:25:75:91:de:51, length 28
13:34:36.945997 02:fb:4f:2a:5f:c9 > 5e:25:75:91:de:51, ethertype ARP (0x0806), length 42: Reply 10.0.1.1 is-at
 02:fb:4f:2a:5f:c9, length 28
```

使用 ping 命令的终端可以看到 ping 的返回信息。



```
[root@simple ~]# ip netns exec ns-tap1 ping -c 1 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=63 time=0.255 ms

--- 10.0.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.255/0.255/0.255/0.000 ms
```

(3) 配置 SNAT

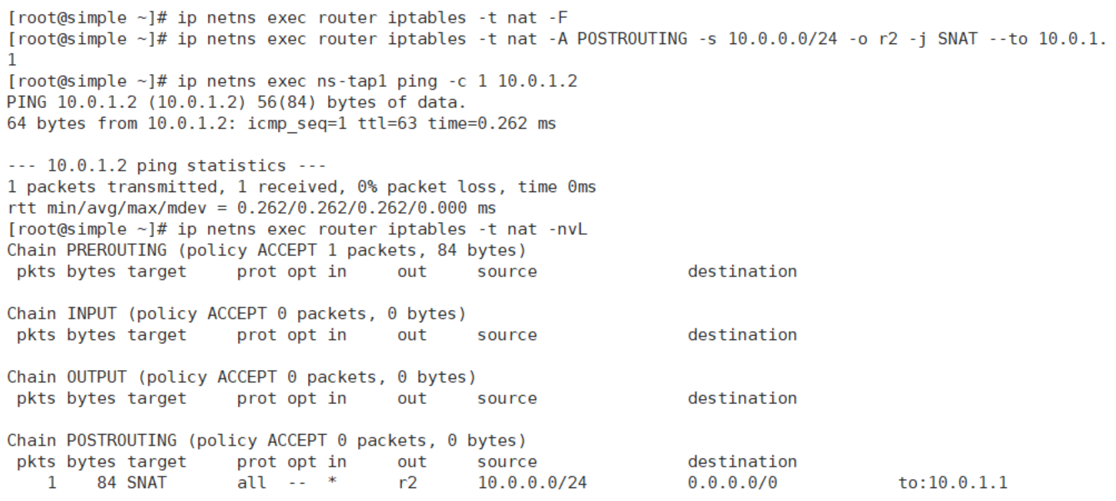
配置 SNAT，然后使用 ping 命令测试连通性，并查看配置的规则信息。

ip netns exec router iptables -t nat -F

ip netns exec router iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o r2 -j SNAT --to 10.0.1.1

ip netns exec ns-tap1 ping -c 1 10.0.1.2

ip netns exec router iptables -t nat -nvL



```
[root@simple ~]# ip netns exec router iptables -t nat -F
[root@simple ~]# ip netns exec router iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o r2 -j SNAT --to 10.0.1.
1
[root@simple ~]# ip netns exec ns-tap1 ping -c 1 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=63 time=0.262 ms

--- 10.0.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.262/0.262/0.262/0.000 ms
[root@simple ~]# ip netns exec router iptables -t nat -nvL
Chain PREROUTING (policy ACCEPT 1 packets, 84 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1    84 SNAT       all  --  *      r2      10.0.0.0/24          0.0.0.0/0            to:10.0.1.1
```

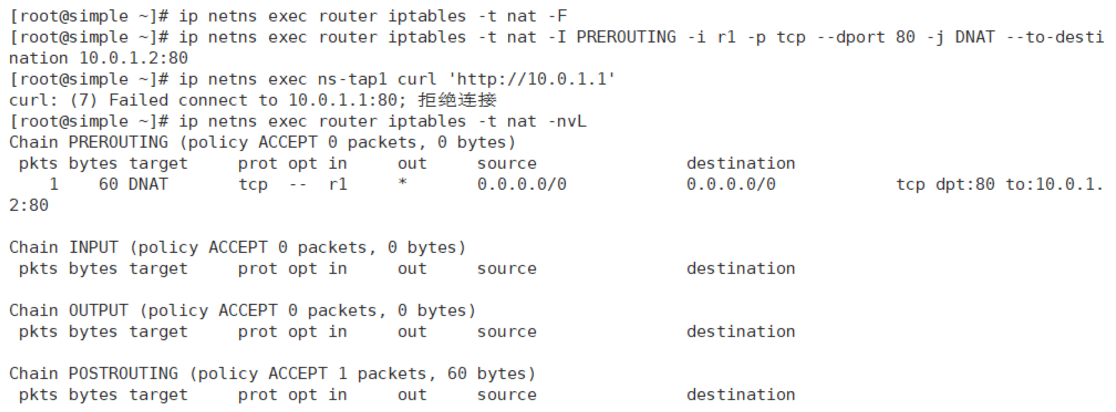
(4) 配置 DNAT

配置 DNAT，并测试是否能够访问 10.0.1.1 的 80 端口，即 http 服务端口，因为本实验机未使用 80 端口，所以返回结果是拒绝连接。

ip netns exec router iptables -t nat -F

ip netns exec router iptables -t nat -I PREROUTING -i r1 -p tcp --dport 80 -j DNAT --to-destination 10.0.1.2:80

ip netns exec ns-tap1 curl http://10.0.1.1



```
[root@simple ~]# ip netns exec router iptables -t nat -F
[root@simple ~]# ip netns exec router iptables -t nat -I PREROUTING -i r1 -p tcp --dport 80 -j DNAT --to-desti
nation 10.0.1.2:80
[root@simple ~]# ip netns exec ns-tap1 curl 'http://10.0.1.1'
curl: (7) Failed connect to 10.0.1.1:80; 拒绝连接
[root@simple ~]# ip netns exec router iptables -t nat -nvL
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1    60 DNAT       tcp  --  r1     *       0.0.0.0/0            0.0.0.0/0            tcp dpt:80 to:10.0.1.
2:80

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 1 packets, 60 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

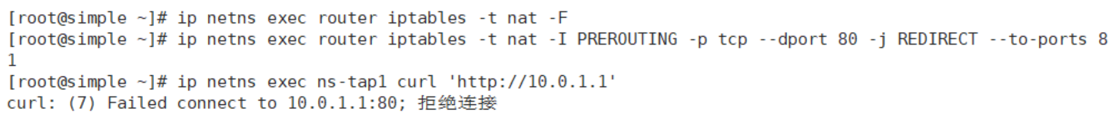
(5) 重定向

设置重定向，再使用 curl 命令测试是否能够访问目标端口。

ip netns exec router iptables -t nat -F

ip netns exec router iptables -t nat -I PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 81

ip netns exec ns-tap1 curl http://10.0.1.1



```
[root@simple ~]# ip netns exec router iptables -t nat -F
[root@simple ~]# ip netns exec router iptables -t nat -I PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8
1
[root@simple ~]# ip netns exec ns-tap1 curl 'http://10.0.1.1'
curl: (7) Failed connect to 10.0.1.1:80; 拒绝连接
```

### 5.3 网络防火墙

(1) 允许内网 1 访问内网 2

设置路由器使得源地址来自内网 1、目标地址为内网 2 的数据包允许通行，再查看已设置的规则信息，最后使用 ping 命令，可以看到内网 1 能够访问内网 2，说明设置的规则生效。

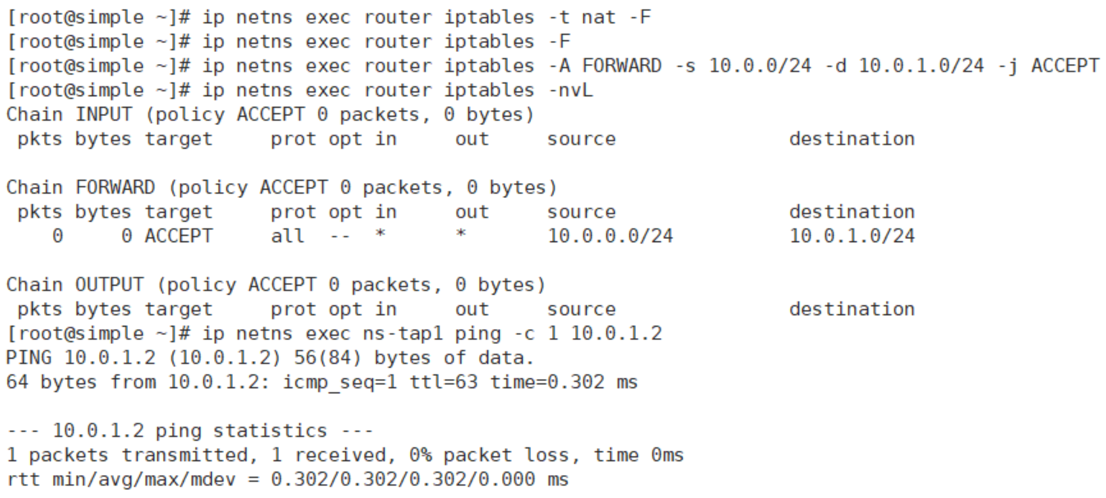
ip netns exec router iptables -t nat -F

ip netns exec router iptables -F

ip netns exec router iptables -A FORWARD -s 10.0.0/24 -d 10.0.1.0/24 -j ACCEPT

ip netns exec router iptables -nvL

ip netns exec ns-tap1 ping -c 1 10.0.1.2



```
[root@simple ~]# ip netns exec router iptables -t nat -F
[root@simple ~]# ip netns exec router iptables -F
[root@simple ~]# ip netns exec router iptables -A FORWARD -s 10.0.0/24 -d 10.0.1.0/24 -j ACCEPT
[root@simple ~]# ip netns exec router iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       10.0.0.0/24          10.0.1.0/24

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
[root@simple ~]# ip netns exec ns-tap1 ping -c 1 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=63 time=0.302 ms

--- 10.0.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.302/0.302/0.302/0.000 ms
```

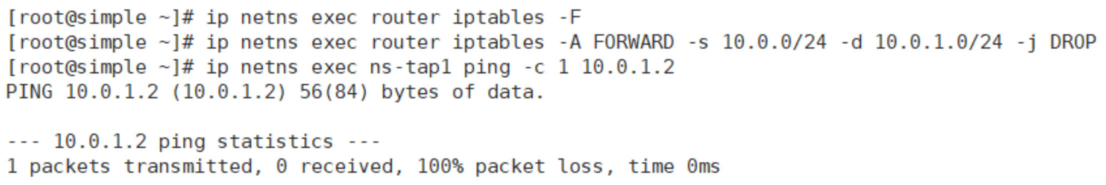
(2) 拒绝内网 1 访问内网 2

设置路由器使得源地址来自内网 1、目标地址为内网 2 的数据包被拦截，再使用 ping 命令可以发现内网 1 无法访问内网 2，说明设置的规则生效。

ip netns exec router iptables -F

ip netns exec router iptables -A FORWARD -s 10.0.0/24 -d 10.0.1.0/24 -j DROP

ip netns exec ns-tap1 ping -c 1 10.0.1.2



```
[root@simple ~]# ip netns exec router iptables -F
[root@simple ~]# ip netns exec router iptables -A FORWARD -s 10.0.0/24 -d 10.0.1.0/24 -j DROP
[root@simple ~]# ip netns exec ns-tap1 ping -c 1 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.

--- 10.0.1.2 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

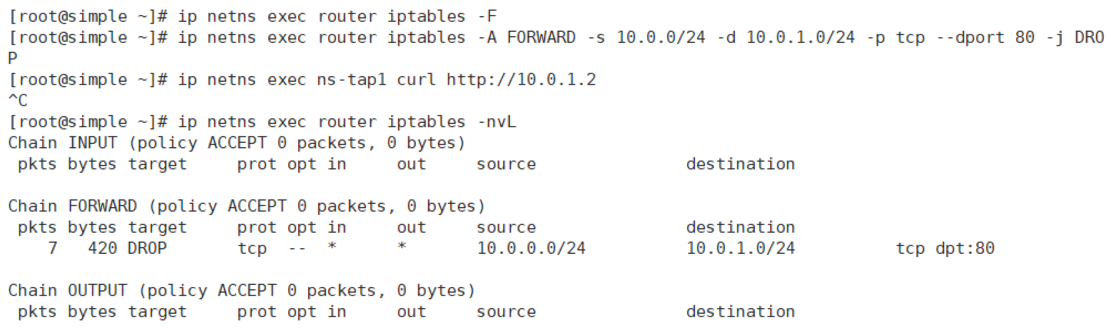
(3) 拒绝内网 1 访问内网 2 的 80 服务

设置路由器使得源地址来自内网 1、目标地址为内网 2 且目标端口是 80 的数据包被拦截，再使用 curl 命令可以发现无返回结果。

ip netns exec router iptables -F

ip netns exec router iptables -A FORWARD -s 10.0.0/24 -d 10.0.1.0/24 -p tcp --dport 80 -j DROP

ip netns exec ns-tap1 curl http://10.0.1.2

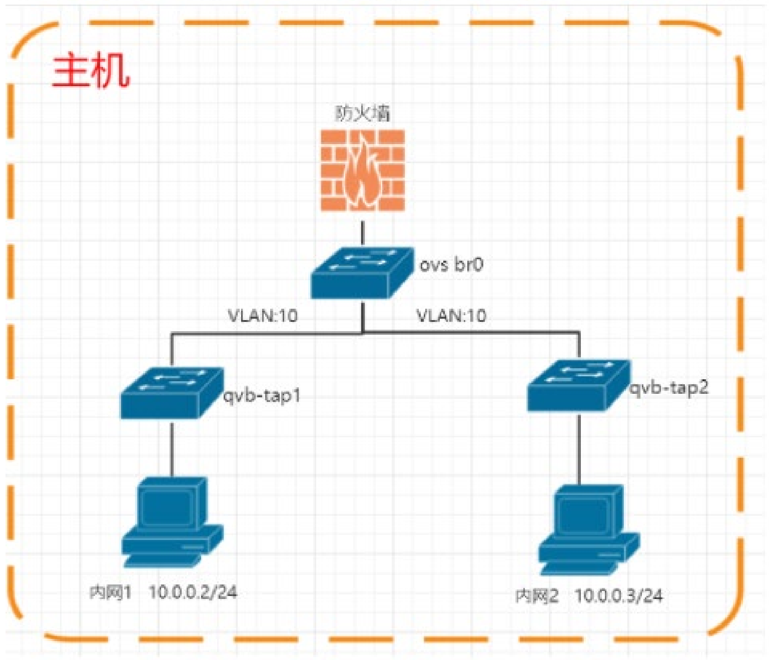


```
[root@simple ~]# ip netns exec router iptables -F
[root@simple ~]# ip netns exec router iptables -A FORWARD -s 10.0.0/24 -d 10.0.1.0/24 -p tcp --dport 80 -j DRO
P
[root@simple ~]# ip netns exec ns-tap1 curl http://10.0.1.2
^C
[root@simple ~]# ip netns exec router iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    7   420 DROP       tcp  --  *      *       10.0.0.0/24          10.0.1.0/24          tcp dpt:80

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

## 任务六

### 6.1 实验网络修改

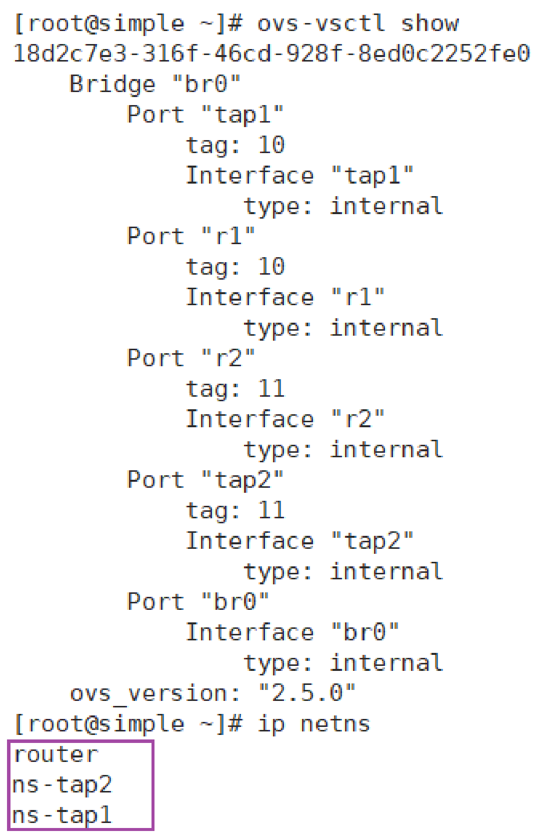
(1) 实验拓扑



(2) 查看上个实验环境

ovs-vsctl show

ip netns



```
[root@simple ~]# ovs-vsctl show
18d2c7e3-316f-46cd-928f-8ed0c2252fe0
    Bridge "br0"
        Port "tap1"
            tag: 10
            Interface "tap1"
                type: internal
        Port "r1"
            tag: 10
            Interface "r1"
                type: internal
        Port "r2"
            tag: 11
            Interface "r2"
                type: internal
        Port "tap2"
            tag: 11
            Interface "tap2"
                type: internal
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "2.5.0"
[root@simple ~]# ip netns
router
ns-tap2
ns-tap1
```

(3) 清空配置

使用如下命令清空之前所配置的网桥、内网和接口等。

ovs-vsctl del-br br0

ip netns del router

ip netns del ns-tap1

ip netns del ns-tap2

(4) 搭建实验环境

新建网桥 br0。

ovs-vsctl add-br br0

配置内网 1。

ip link add qvo-tap1 type veth peer name qvb-tap1

ip link set qvb-tap1 up

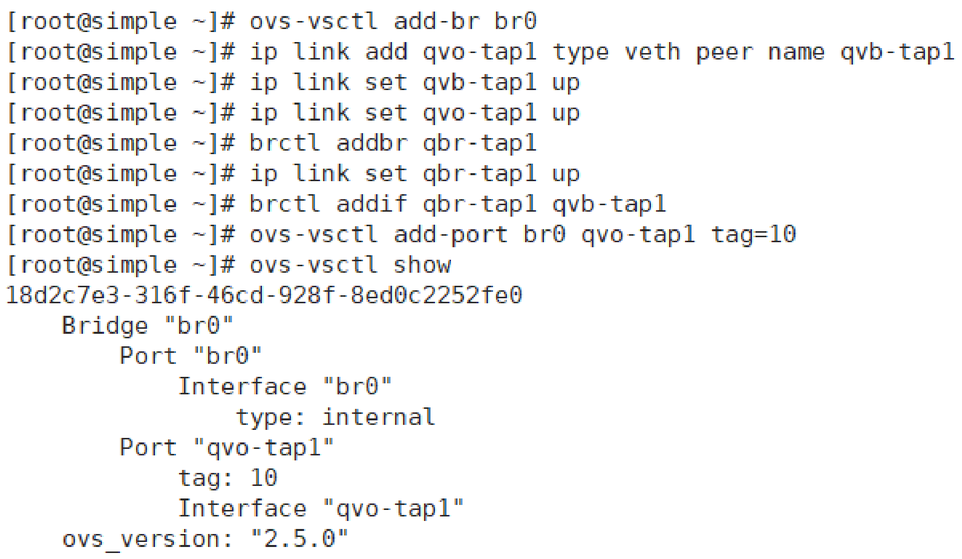
ip link set qvo-tap1 up

brctl addbr qbr-tap1

ip link set qbr-tap1 up

brctl addif qbr-tap1 qvb-tap1

ovs-vsctl add-port br0 qvo-tap1 tag=10



```
[root@simple ~]# ovs-vsctl add-br br0
[root@simple ~]# ip link add qvo-tap1 type veth peer name qvb-tap1
[root@simple ~]# ip link set qvb-tap1 up
[root@simple ~]# ip link set qvo-tap1 up
[root@simple ~]# brctl addbr qbr-tap1
[root@simple ~]# ip link set qbr-tap1 up
[root@simple ~]# brctl addif qbr-tap1 qvb-tap1
[root@simple ~]# ovs-vsctl add-port br0 qvo-tap1 tag=10
[root@simple ~]# ovs-vsctl show
18d2c7e3-316f-46cd-928f-8ed0c2252fe0
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
        Port "qvo-tap1"
            tag: 10
            Interface "qvo-tap1"
    ovs_version: "2.5.0"
```

ip link add tap1 type veth peer name tap11

brctl addif qbr-tap1 tap11

ip link set tap11 up

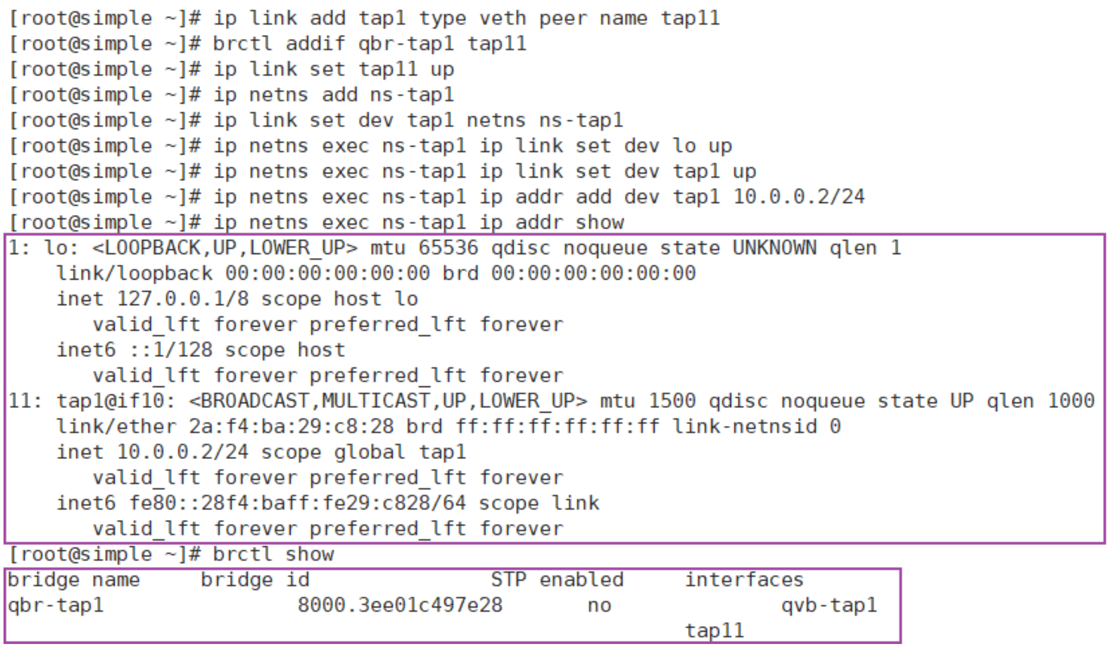
ip netns add ns-tap1

ip link set dev tap1 netns ns-tap1

ip netns exec ns-tap1 ip link set dev lo up

ip netns exec ns-tap1 ip link set dev tap1 up

ip netns exec ns-tap1 ip addr add dev tap1 10.0.0.2/24



```
[root@simple ~]# ip link add tap1 type veth peer name tap11
[root@simple ~]# brctl addif qbr-tap1 tap11
[root@simple ~]# ip link set tap11 up
[root@simple ~]# ip netns add ns-tap1
[root@simple ~]# ip link set dev tap1 netns ns-tap1
[root@simple ~]# ip netns exec ns-tap1 ip link set dev lo up
[root@simple ~]# ip netns exec ns-tap1 ip link set dev tap1 up
[root@simple ~]# ip netns exec ns-tap1 ip addr add dev tap1 10.0.0.2/24
[root@simple ~]# ip netns exec ns-tap1 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
11: tap1@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP qlen 1000
    link/ether 2a:f4:ba:29:c8:28 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.2/24 scope global tap1
       valid_lft forever preferred_lft forever
    inet6 fe80::28f4:baff:fe29:c828/64 scope link
       valid_lft forever preferred_lft forever
[root@simple ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
qbr-tap1                8000.3ee01c497e28       no              qvb-tap1
                                                        tap11
```

配置内网 2。

ip link add qvo-tap2 type veth peer name qvb-tap2

ip link set qvb-tap2 up

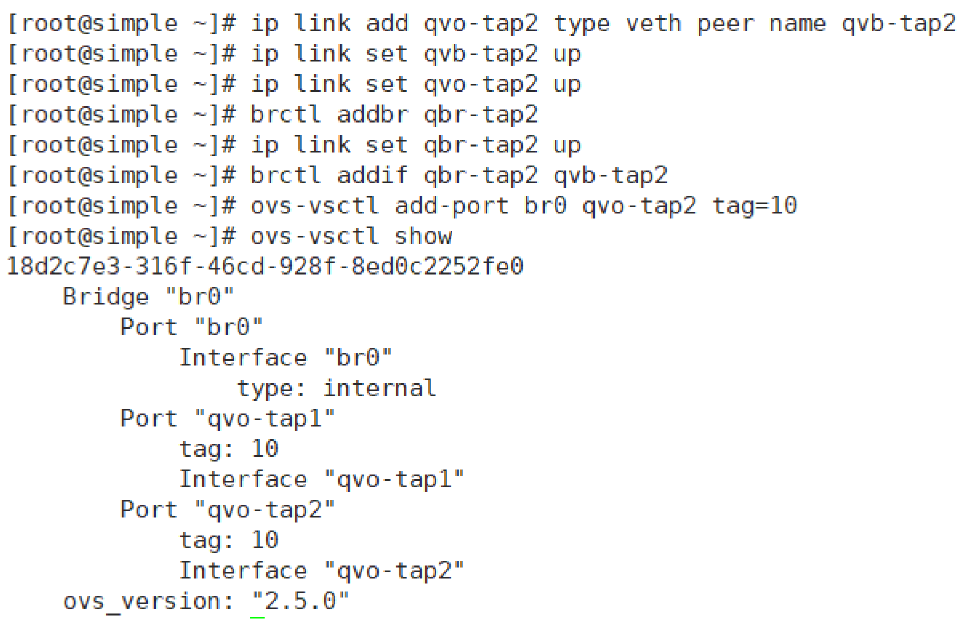
ip link set qvo-tap2 up

brctl addbr qbr-tap2

ip link set qbr-tap2 up

brctl addif qbr-tap2 qvb-tap2

ovs-vsctl add-port br0 qvo-tap2 tag=10



```
[root@simple ~]# ip link add qvo-tap2 type veth peer name qvb-tap2
[root@simple ~]# ip link set qvb-tap2 up
[root@simple ~]# ip link set qvo-tap2 up
[root@simple ~]# brctl addbr qbr-tap2
[root@simple ~]# ip link set qbr-tap2 up
[root@simple ~]# brctl addif qbr-tap2 qvb-tap2
[root@simple ~]# ovs-vsctl add-port br0 qvo-tap2 tag=10
[root@simple ~]# ovs-vsctl show
18d2c7e3-316f-46cd-928f-8ed0c2252fe0
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
        Port "qvo-tap1"
            tag: 10
            Interface "qvo-tap1"
        Port "qvo-tap2"
            tag: 10
            Interface "qvo-tap2"
    ovs_version: "2.5.0"
```

ip link add tap2 type veth peer name tap22

brctl addif qbr-tap2 tap22

ip link set tap22 up

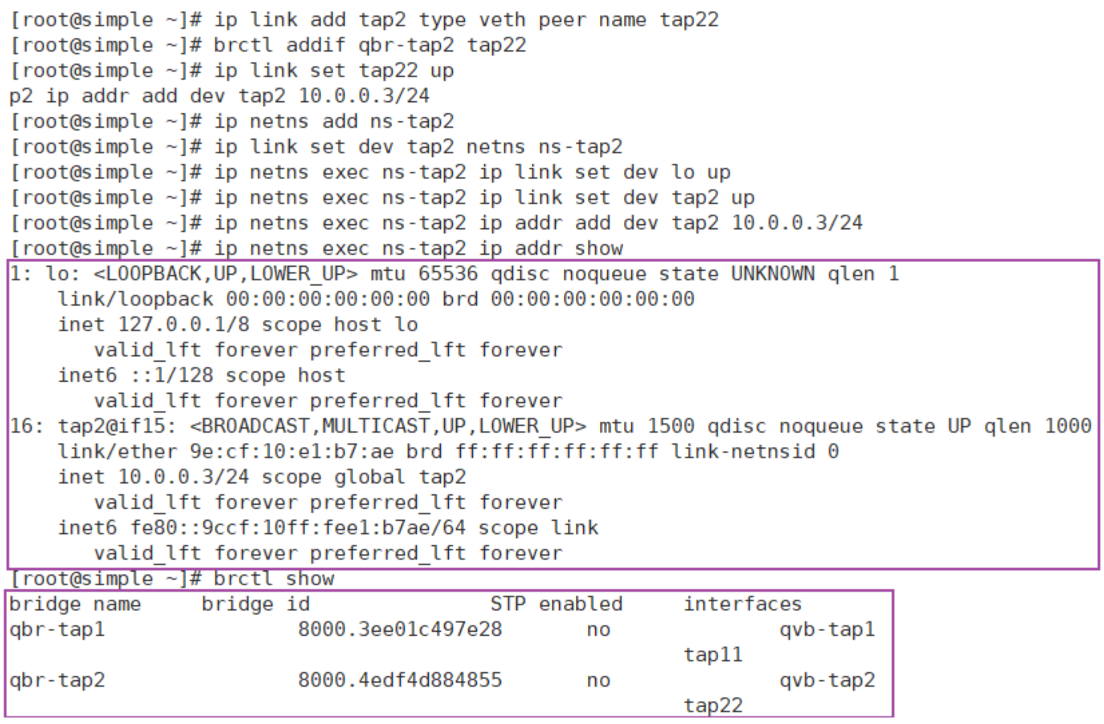
ip netns add ns-tap2

ip link set dev tap2 netns ns-tap2

ip netns exec ns-tap2 ip link set dev lo up

ip netns exec ns-tap2 ip link set dev tap2 up

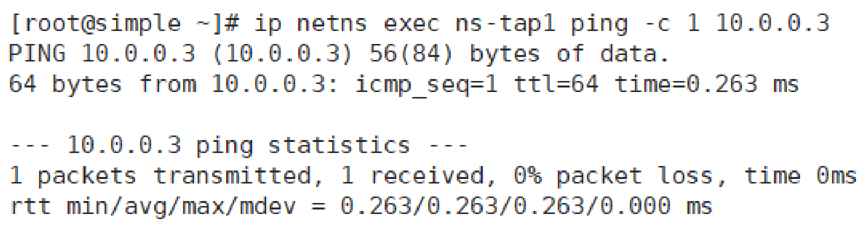
ip netns exec ns-tap2 ip addr add dev tap2 10.0.0.3/24



```
[root@simple ~]# ip link add tap2 type veth peer name tap22
[root@simple ~]# brctl addif qbr-tap2 tap22
[root@simple ~]# ip link set tap22 up
p2 ip addr add dev tap2 10.0.0.3/24
[root@simple ~]# ip netns add ns-tap2
[root@simple ~]# ip link set dev tap2 netns ns-tap2
[root@simple ~]# ip netns exec ns-tap2 ip link set dev lo up
[root@simple ~]# ip netns exec ns-tap2 ip link set dev tap2 up
[root@simple ~]# ip netns exec ns-tap2 ip addr add dev tap2 10.0.0.3/24
[root@simple ~]# ip netns exec ns-tap2 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
16: tap2@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP qlen 1000
    link/ether 9e:cf:10:e1:b7:ae brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.3/24 scope global tap2
       valid_lft forever preferred_lft forever
    inet6 fe80::9ccf:10ff:fee1:b7ae/64 scope link
       valid_lft forever preferred_lft forever
[root@simple ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
qbr-tap1                8000.3ee01c497e28       no              qvb-tap1
                                                        tap11
qbr-tap2                8000.4edf4d884855       no              qvb-tap2
                                                        tap22
```

测试二层同网段通信。

ip netns exec ns-tap1 ping -c 1 10.0.0.3

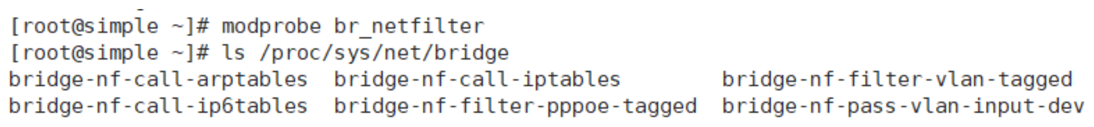


```
[root@simple ~]# ip netns exec ns-tap1 ping -c 1 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.263 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.263/0.263/0.263/0.000 ms
```

(5) 加载内核参数

modprobe br_netfilter

ls /proc/sys/net/bridge



```
[root@simple ~]# modprobe br_netfilter
[root@simple ~]# ls /proc/sys/net/bridge
bridge-nf-call-arptables  bridge-nf-call-iptables      bridge-nf-filter-vlan-tagged
bridge-nf-call-ip6tables  bridge-nf-filter-pppoe-tagged  bridge-nf-pass-vlan-input-dev
```

需要开启 bridge-nf，使得数据不会直接经过网桥转发，从而使得 FORWARD 链设置生效。

echo "net.bridge.bridge-nf-call-arptables = 1" >> /etc/sysctl.conf

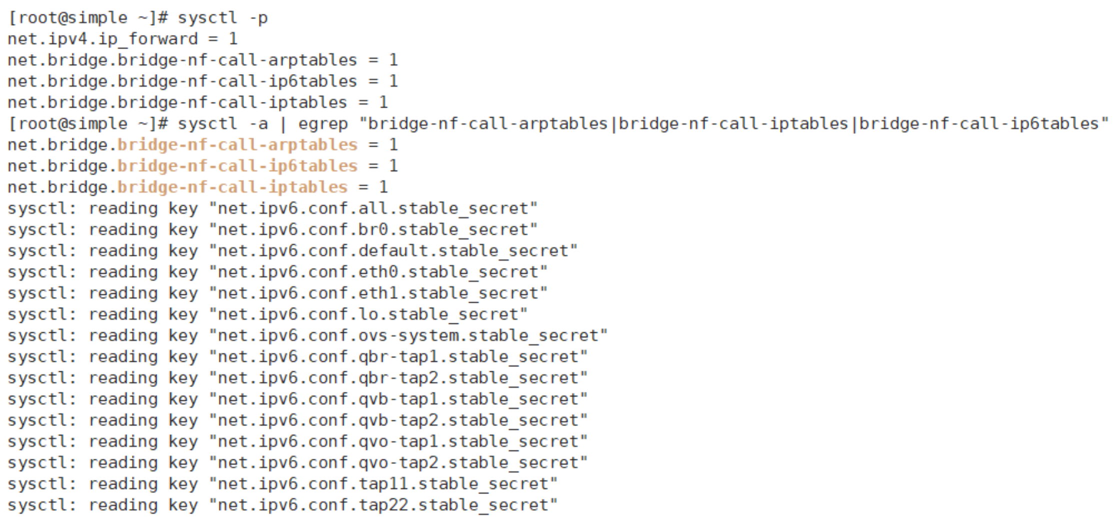
echo "net.bridge.bridge-nf-call-ip6tables = 1" >> /etc/sysctl.conf

echo "net.bridge.bridge-nf-call-iptables = 1" >> /etc/sysctl.conf

将设置项写入配置文件后，使用以下命令使得设置生效。

sysctl -p

sysctl -a | egrep "bridge-nf-call-arptables|bridge-nf-call-iptables|bridge-nf-call-ip6tables"



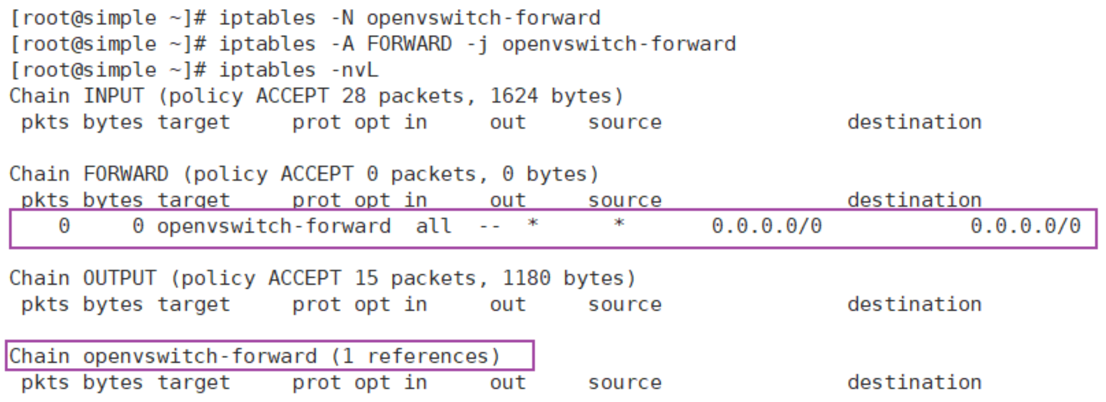
```
[root@simple ~]# sysctl -p
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-arptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
[root@simple ~]# sysctl -a | egrep "bridge-nf-call-arptables|bridge-nf-call-iptables|bridge-nf-call-ip6tables"
net.bridge.bridge-nf-call-arptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
sysctl: reading key "net.ipv6.conf.all.stable_secret"
sysctl: reading key "net.ipv6.conf.br0.stable_secret"
sysctl: reading key "net.ipv6.conf.default.stable_secret"
sysctl: reading key "net.ipv6.conf.eth0.stable_secret"
sysctl: reading key "net.ipv6.conf.eth1.stable_secret"
sysctl: reading key "net.ipv6.conf.lo.stable_secret"
sysctl: reading key "net.ipv6.conf.ovs-system.stable_secret"
sysctl: reading key "net.ipv6.conf.qbr-tap1.stable_secret"
sysctl: reading key "net.ipv6.conf.qbr-tap2.stable_secret"
sysctl: reading key "net.ipv6.conf.qvb-tap1.stable_secret"
sysctl: reading key "net.ipv6.conf.qvb-tap2.stable_secret"
sysctl: reading key "net.ipv6.conf.qvo-tap1.stable_secret"
sysctl: reading key "net.ipv6.conf.qvo-tap2.stable_secret"
sysctl: reading key "net.ipv6.conf.tap11.stable_secret"
sysctl: reading key "net.ipv6.conf.tap22.stable_secret"
```

(6) 二层防火墙配置

把 FORWARD 链所有流量导入自定义链。

iptables -N openvswitch-forward

iptables -A FORWARD -j openvswitch-forward



```
[root@simple ~]# iptables -N openvswitch-forward
[root@simple ~]# iptables -A FORWARD -j openvswitch-forward
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 openvswitch-forward  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain openvswitch-forward (1 references)
 pkts bytes target     prot opt in     out     source               destination
```

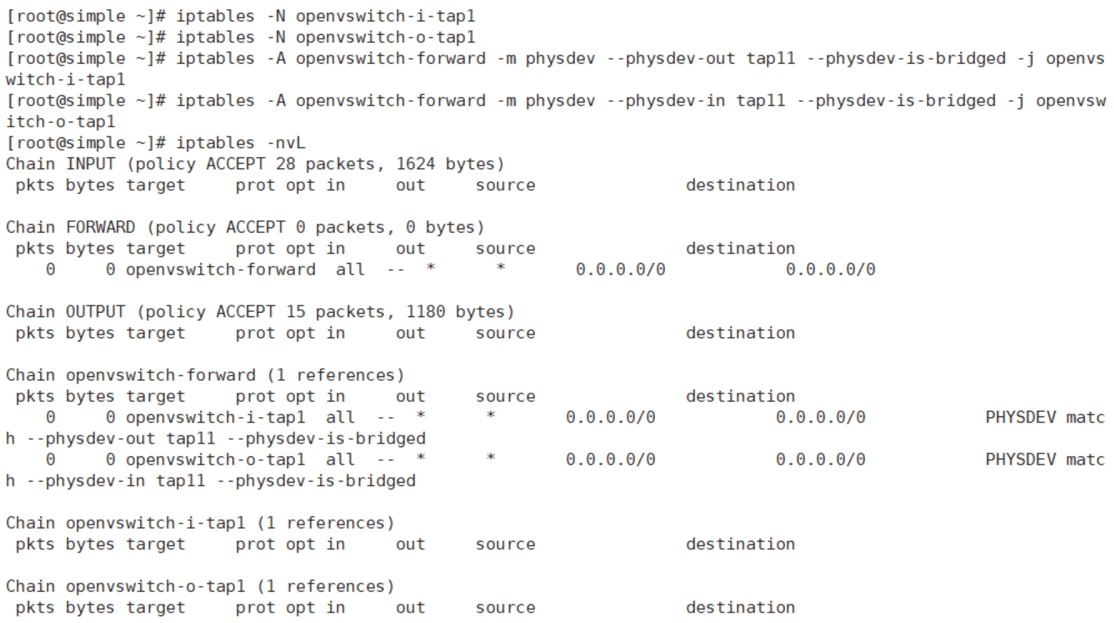
添加内网 1 in、out 方向链表，将匹配到的 physdev 流量倒入进出链表。

iptables -N openvswitch-i-tap1

iptables -N openvswitch-o-tap1

iptables -A openvswitch-forward -m physdev --physdev-out tap11 --physdev-is-bridged -j openvswitch-i-tap1

iptables -A openvswitch-forward -m physdev --physdev-in tap11 --physdev-is-bridged -j openvswitch-o-tap1



```
[root@simple ~]# iptables -N openvswitch-i-tap1
[root@simple ~]# iptables -N openvswitch-o-tap1
[root@simple ~]# iptables -A openvswitch-forward -m physdev --physdev-out tap11 --physdev-is-bridged -j openvs
witch-i-tap1
[root@simple ~]# iptables -A openvswitch-forward -m physdev --physdev-in tap11 --physdev-is-bridged -j openvsw
itch-o-tap1
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 openvswitch-forward  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain openvswitch-forward (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 openvswitch-i-tap1  all  --  *      *       0.0.0.0/0            0.0.0.0/0            PHYSDEV matc
h --physdev-out tap11 --physdev-is-bridged
    0     0 openvswitch-o-tap1  all  --  *      *       0.0.0.0/0            0.0.0.0/0            PHYSDEV matc
h --physdev-in tap11 --physdev-is-bridged

Chain openvswitch-i-tap1 (1 references)
 pkts bytes target     prot opt in     out     source               destination

Chain openvswitch-o-tap1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
```

添加内网 1 in 方向规则，并将没有匹配的流量导入新链表，过滤源地址使用

ipset 管理。

iptables -N openvswitch-fallback

ipset create ipv4-tap1 hash:net

iptables -A openvswitch-i-tap1 -m set --match-set ipv4-tap1 src -j RETURN

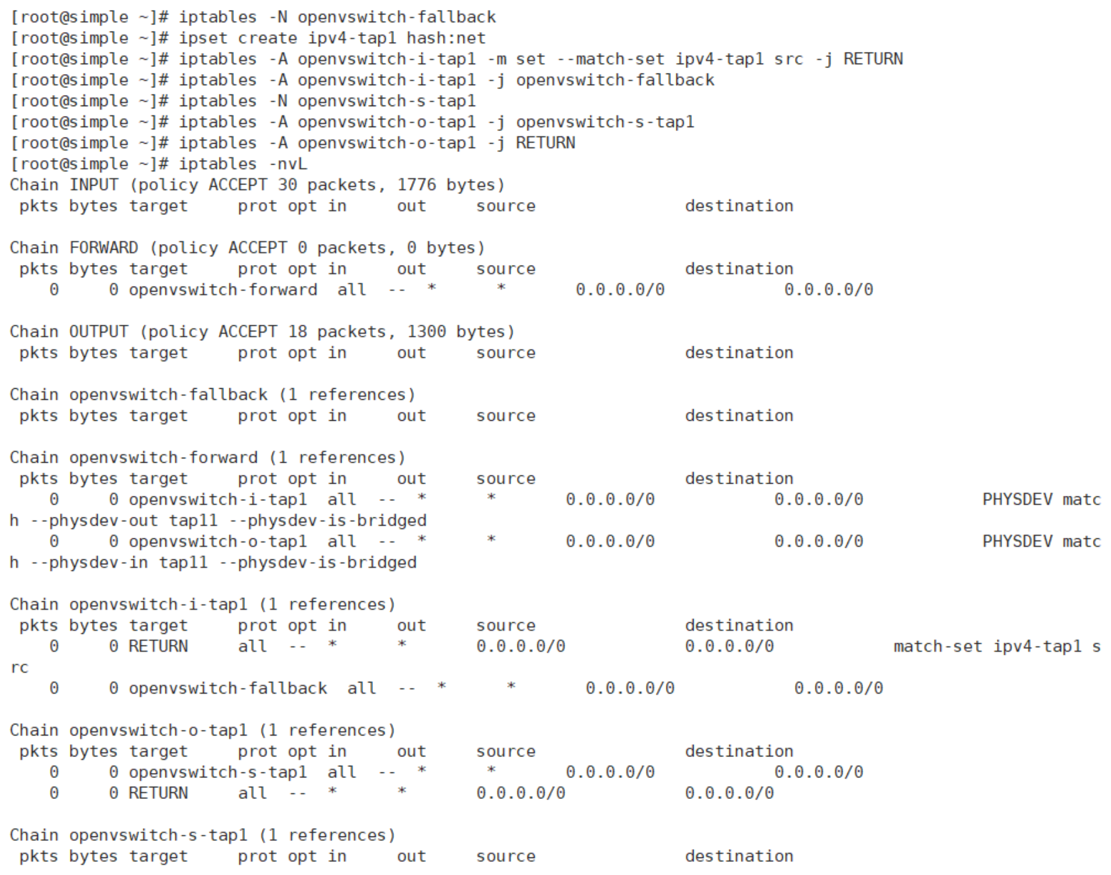
iptables -A openvswitch-i-tap1 -j openvswitch-fallback

添加内网 1 out 方向规则，将所有流量导入新的链过滤。

iptables -N openvswitch-s-tap1

iptables -A openvswitch-o-tap1 -j openvswitch-s-tap1

iptables -A openvswitch-o-tap1 -j RETURN



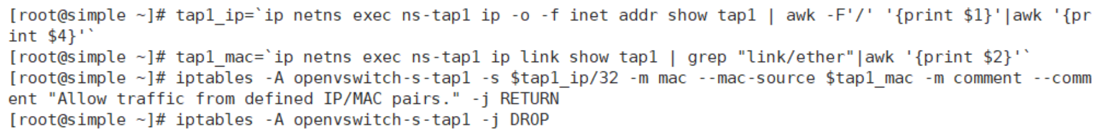
```
[root@simple ~]# iptables -N openvswitch-fallback
[root@simple ~]# ipset create ipv4-tap1 hash:net
[root@simple ~]# iptables -A openvswitch-i-tap1 -m set --match-set ipv4-tap1 src -j RETURN
[root@simple ~]# iptables -A openvswitch-i-tap1 -j openvswitch-fallback
[root@simple ~]# iptables -N openvswitch-s-tap1
[root@simple ~]# iptables -A openvswitch-o-tap1 -j openvswitch-s-tap1
[root@simple ~]# iptables -A openvswitch-o-tap1 -j RETURN
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 30 packets, 1776 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 openvswitch-forward  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 18 packets, 1300 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain openvswitch-fallback (1 references)
 pkts bytes target     prot opt in     out     source               destination

Chain openvswitch-forward (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 openvswitch-i-tap1  all  --  *      *       0.0.0.0/0            0.0.0.0/0            PHYSDEV matc
h --physdev-out tap11 --physdev-is-bridged
    0     0 openvswitch-o-tap1  all  --  *      *       0.0.0.0/0            0.0.0.0/0            PHYSDEV matc
h --physdev-in tap11 --physdev-is-bridged

Chain openvswitch-i-tap1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0            match-set ipv4-tap1 s
rc
    0     0 openvswitch-fallback  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain openvswitch-o-tap1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 openvswitch-s-tap1  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain openvswitch-s-tap1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
```

添加内网 1 安全规则。

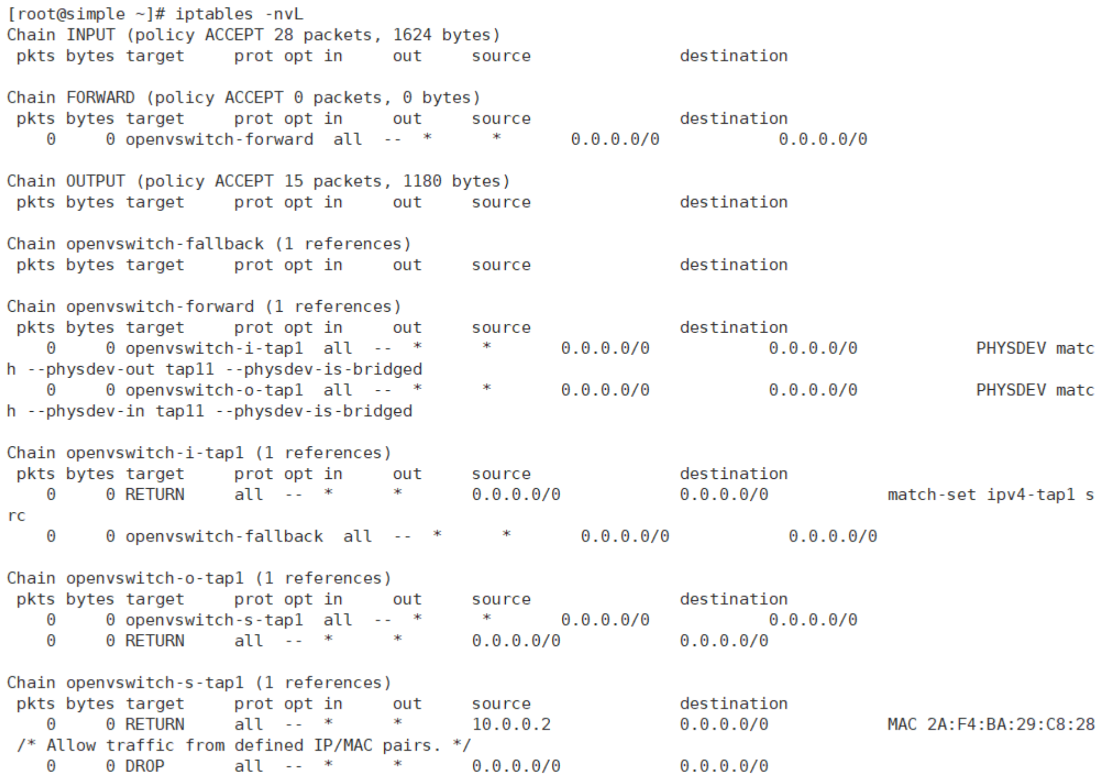
tap1_ip=`ip netns exec ns-tap1 ip -o -f inet addr show tap1 | awk -F/ {print $1}|awk {print $4}`

tap1_mac=`ip netns exec ns-tap1 ip link show tap1 | grep "link/ether"|awk {print $2}`

iptables -A openvswitch-s-tap1 -s $tap1_ip/32 -m mac --mac-source $tap1_mac -m comment --comment "Allow traffic from defined IP/MAC pairs." -j RETURN iptables -A openvswitch-s-tap1 -j DROP



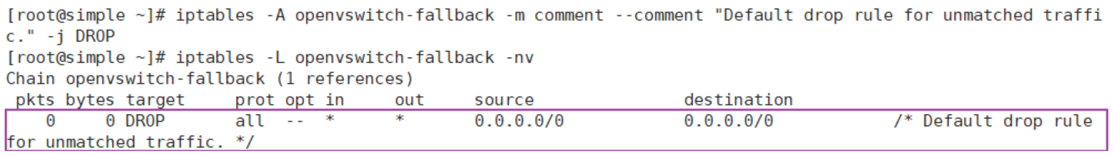
```
[root@simple ~]# tap1_ip=`ip netns exec ns-tap1 ip -o -f inet addr show tap1 | awk -F'/' '{print $1}'|awk '{pr
int $4}'`
[root@simple ~]# tap1_mac=`ip netns exec ns-tap1 ip link show tap1 | grep "link/ether"|awk '{print $2}'`
[root@simple ~]# iptables -A openvswitch-s-tap1 -s $tap1_ip/32 -m mac --mac-source $tap1_mac -m comment --comm
ent "Allow traffic from defined IP/MAC pairs." -j RETURN
[root@simple ~]# iptables -A openvswitch-s-tap1 -j DROP
```



```
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 28 packets, 1624 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 openvswitch-forward  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 15 packets, 1180 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain openvswitch-fallback (1 references)
 pkts bytes target     prot opt in     out     source               destination

Chain openvswitch-forward (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 openvswitch-i-tap1  all  --  *      *       0.0.0.0/0            0.0.0.0/0            PHYSDEV matc
h --physdev-out tap11 --physdev-is-bridged
    0     0 openvswitch-o-tap1  all  --  *      *       0.0.0.0/0            0.0.0.0/0            PHYSDEV matc
h --physdev-in tap11 --physdev-is-bridged

Chain openvswitch-i-tap1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0            match-set ipv4-tap1 s
rc
    0     0 openvswitch-fallback  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain openvswitch-o-tap1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 openvswitch-s-tap1  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain openvswitch-s-tap1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 RETURN     all  --  *      *       10.0.0.2             0.0.0.0/0            MAC 2A:F4:BA:29:C8:28
 /* Allow traffic from defined IP/MAC pairs. */
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

拒绝未匹配的流量。

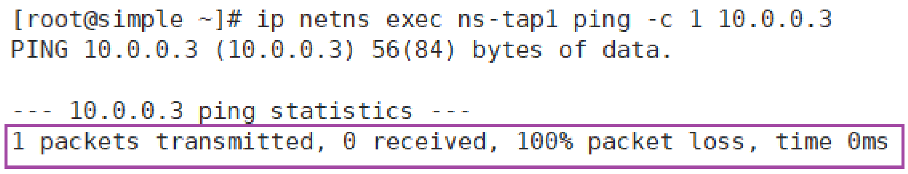
iptables -A openvswitch-fallback -m comment --comment "Default drop rule for unmatched traffic." -j DROP



```
[root@simple ~]# iptables -A openvswitch-fallback -m comment --comment "Default drop rule for unmatched traffi
c." -j DROP
[root@simple ~]# iptables -L openvswitch-fallback -nv
Chain openvswitch-fallback (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            /* Default drop rule
for unmatched traffic. */
```

测试内网 1 与内网 2 的网络连通性。可以发现此时内网 1 无法 ping 通内网 2，说明二层防火墙策略生效。

ip netns exec ns-tap1 ping -c 1 10.0.0.3



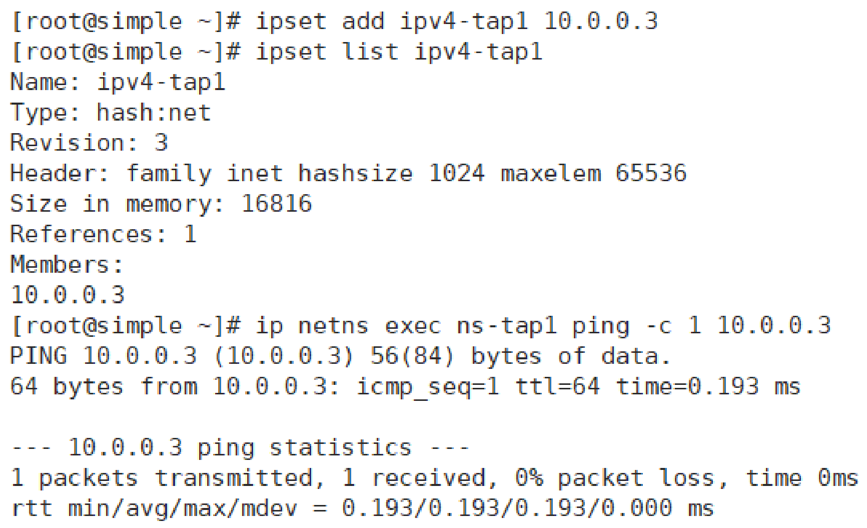
```
[root@simple ~]# ip netns exec ns-tap1 ping -c 1 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

ipset 添加源地址内网 2，验证连通性。

ipset add ipv4-tap1 10.0.0.3

ipset list ipv4-tap1

ip netns exec ns-tap1 ping -c 1 10.0.0.3



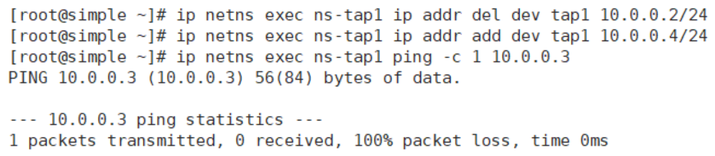
```
[root@simple ~]# ipset add ipv4-tap1 10.0.0.3
[root@simple ~]# ipset list ipv4-tap1
Name: ipv4-tap1
Type: hash:net
Revision: 3
Header: family inet hashsize 1024 maxelem 65536
Size in memory: 16816
References: 1
Members:
10.0.0.3
[root@simple ~]# ip netns exec ns-tap1 ping -c 1 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.193 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.193/0.193/0.193/0.000 ms
```

验证内网 1 能否修改 IP、MAC。

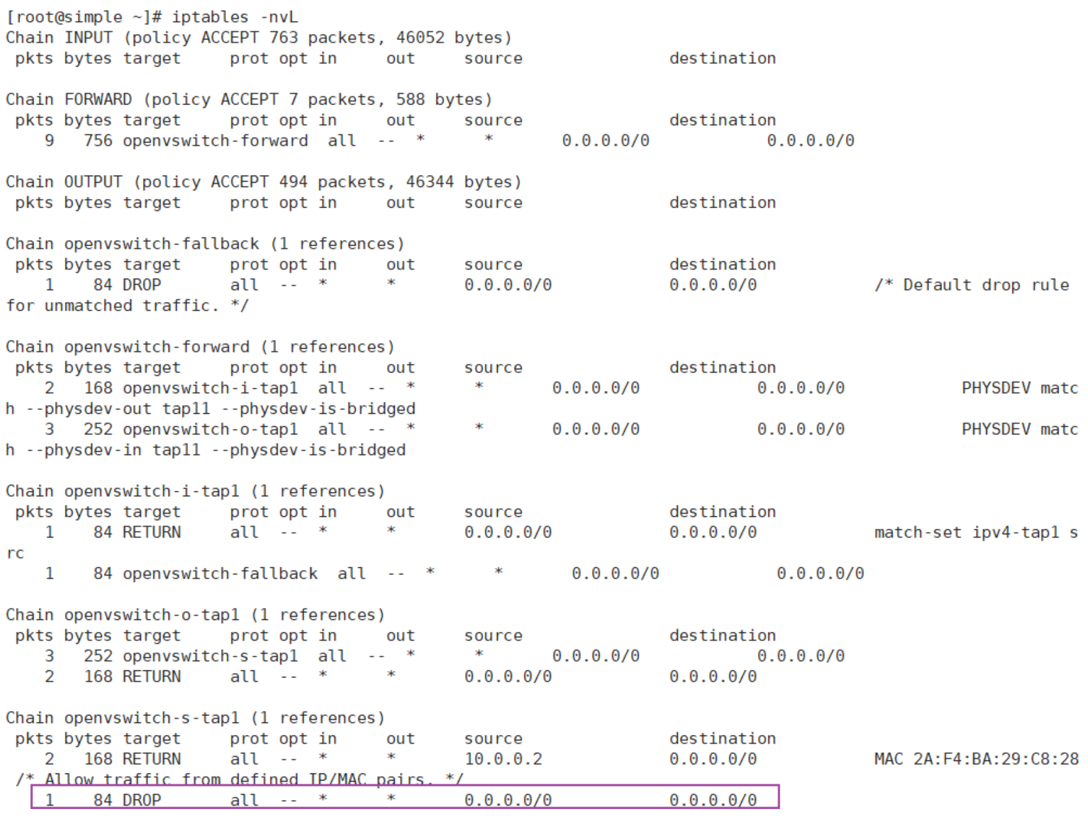
使用如下命令验证内网 1 能否修改 IP，可以发现实验中的策略可以防止用户随意修改 IP 地址。

ip netns exec ns-tap1 ip addr del dev tap1 10.0.0.2/24

ip netns exec ns-tap1 ip addr add dev tap1 10.0.0.4/24



```
[root@simple ~]# ip netns exec ns-tap1 ip addr del dev tap1 10.0.0.2/24
[root@simple ~]# ip netns exec ns-tap1 ip addr add dev tap1 10.0.0.4/24
[root@simple ~]# ip netns exec ns-tap1 ping -c 1 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```



```
[root@simple ~]# iptables -nvL
Chain INPUT (policy ACCEPT 763 packets, 46052 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 7 packets, 588 bytes)
 pkts bytes target     prot opt in     out     source               destination
    9   756 openvswitch-forward  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 494 packets, 46344 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain openvswitch-fallback (1 references)
 pkts bytes target     prot opt in     out     source               destination
    1    84 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            /* Default drop rule
for unmatched traffic. */

Chain openvswitch-forward (1 references)
 pkts bytes target     prot opt in     out     source               destination
    2   168 openvswitch-i-tap1  all  --  *      *       0.0.0.0/0            0.0.0.0/0            PHYSDEV matc
h --physdev-out tap11 --physdev-is-bridged
    3   252 openvswitch-o-tap1  all  --  *      *       0.0.0.0/0            0.0.0.0/0            PHYSDEV matc
h --physdev-in tap11 --physdev-is-bridged

Chain openvswitch-i-tap1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    1    84 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0            match-set ipv4-tap1 s
rc
    1    84 openvswitch-fallback  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain openvswitch-o-tap1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    3   252 openvswitch-s-tap1  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    2   168 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain openvswitch-s-tap1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    2   168 RETURN     all  --  *      *       10.0.0.2             0.0.0.0/0            MAC 2A:F4:BA:29:C8:28
 /* Allow traffic from defined IP/MAC pairs. */
    1    84 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

将内网 1 还原为原 IP。

ip netns exec ns-tap1 ip addr del dev tap1 10.0.0.4/24

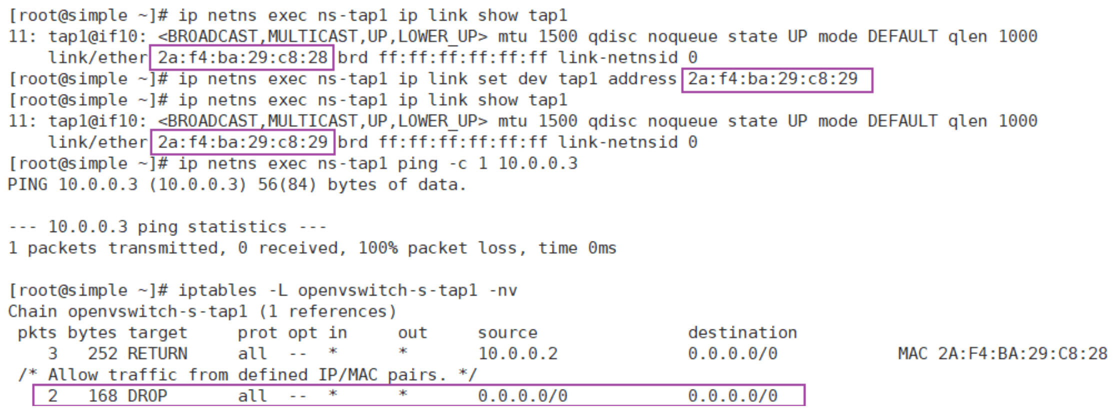
ip netns exec ns-tap1 ip addr add dev tap1 10.0.0.2/24

使用如下命令验证内网 1 能否修改 MAC，可以发现实验中的策略可以防止用户随意修改 MAC 地址。

ip netns exec ns-tap1 ip link set dev tap1 address 2a:f4:ba:29:c8:29

ip netns exec ns-tap1 ping -c 1 10.0.0.3

iptables -L openvswitch-s-tap1 -nv



```
[root@simple ~]# ip netns exec ns-tap1 ip link show tap1
11: tap1@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT qlen 1000
    link/ether 2a:f4:ba:29:c8:28 brd ff:ff:ff:ff:ff:ff link-netnsid 0
[root@simple ~]# ip netns exec ns-tap1 ip link set dev tap1 address 2a:f4:ba:29:c8:29
[root@simple ~]# ip netns exec ns-tap1 ip link show tap1
11: tap1@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT qlen 1000
    link/ether 2a:f4:ba:29:c8:29 brd ff:ff:ff:ff:ff:ff link-netnsid 0
[root@simple ~]# ip netns exec ns-tap1 ping -c 1 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

[root@simple ~]# iptables -L openvswitch-s-tap1 -nv
Chain openvswitch-s-tap1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    3   252 RETURN     all  --  *      *       10.0.0.2             0.0.0.0/0            MAC 2A:F4:BA:29:C8:28
 /* Allow traffic from defined IP/MAC pairs. */
    2   168 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

## 【实验总结】