

IPTABLES 综合实验

本次实验通过设置 iptables 规则，来实现防火墙功能。iptables 本身不是防火墙，它可以理解为一个客户端代理，通过 iptables 来设定规则，最终运行在 netfilter 框架下。

Iptables 中有四链五表，用来过滤所有进出的报文。防火墙的功能是对经过的报文匹配规则，然后执行对应的动作。对于每个关卡，都不止一条规则，故称为链。将那些具有相同规则的集合叫做表。链包括 INPUT、OUTPUT、PREROUTING(路由前)、FORWARD(转发)、POSTROUTING(路由后)。同时，在 iptables 中预定义了四种表，分别是负责过滤的 filter 表、负责网络地址转换的 nat 表、负责拆解报文做出修改的 mangle 表以及关闭 nat 表上启用的连接追踪机制的 raw 表。本次实验中主要涉及了前两个表。

通过本次实验，学习了很多关于 iptables 以及网络、安全方面的知识。我将学到的知识治理成以下三个实验内容：主机防火墙、网络防火墙、NAT 转换。

在理论课的过程中，老师经常提及 Dos 攻击。所以，在学习了 socket 编程之后，用 C 语言实现了两台虚拟机之间建立 TCP 连接。最终模拟了防御 Dos 攻击的情况，形成了实验四。

总的来说，本次实验我递进式的做了四个实验，分别是主机防火墙、网络防火墙、NAT 动作、TCP 连接与 DoS 防御。

实验一 主机防火墙

一、实验目的

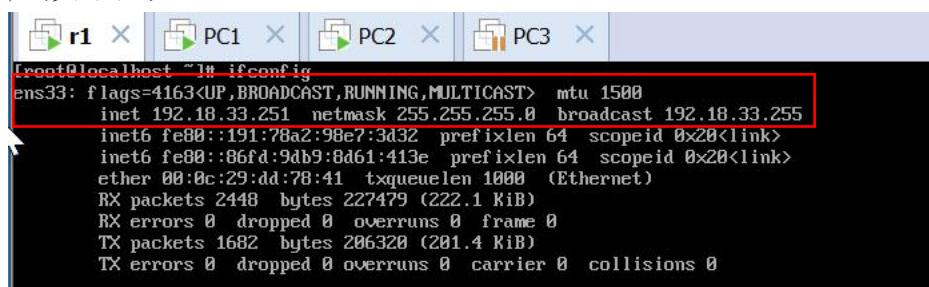
1. 通过配置主机防火墙熟悉 iptables 常用指令。
2. 通过简单的两台虚拟机环境测试设置的规则。

二、实验内容

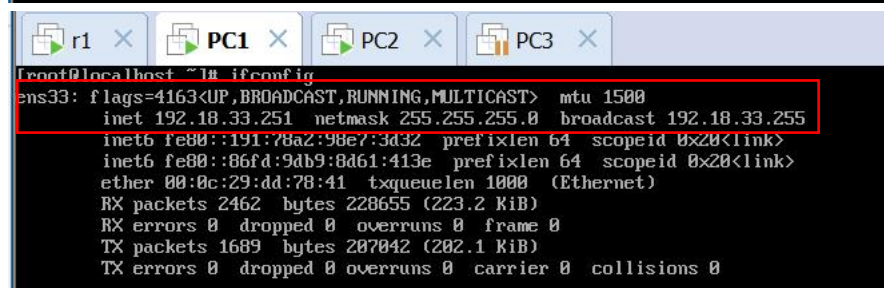
1. 设置白名单机制，该默认为所有端口不许通行，仅允许 80 端口和 22 端口。
2. 设置日志记录 80 端口，22 端口情况。通过另一台主机的 ssh, curl, ping 命令进行测试，观察现象。
3. 通过防火墙拒绝 tcp 第一次握手，来阻止部分想通过 22 端口的包。
4. 通过限制 icmp 四种报文类型中的主机不可达报文来限制 ping 命令。
5. 将 3, 4 通过仅允许响应报文通过防火墙来实现相同效果。

三、实验过程

1. 设置两台虚拟机 R1(192.18.33.251)和 PC1(192.18.33.33)，并配置二者在同一个虚拟网络下。

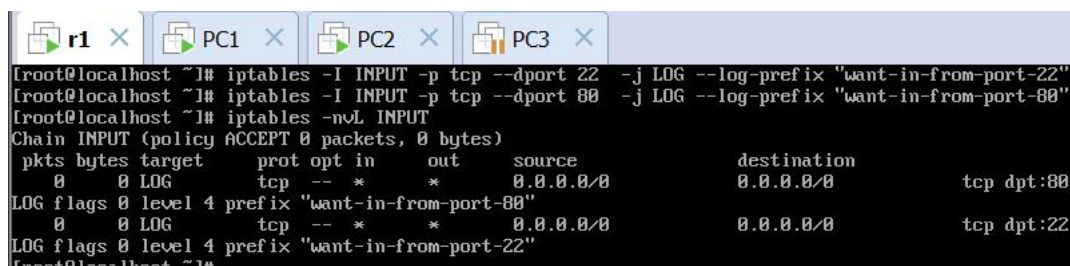


```
root@localhost ~# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.18.33.251 netmask 255.255.255.0 broadcast 192.18.33.255
    inet6 fe80::191:78a2:98e7:3d32 prefixlen 64 scopeid 0x20<link>
    inet6 fe80::86fd:9db9:8d61:413e prefixlen 64 scopeid 0x20<link>
    ether 08:0c:29:dd:78:41 txqueuelen 1000 (Ethernet)
    RX packets 2448 bytes 227479 (222.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1682 bytes 206320 (201.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```



```
root@localhost ~# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.18.33.251 netmask 255.255.255.0 broadcast 192.18.33.255
    inet6 fe80::191:78a2:98e7:3d32 prefixlen 64 scopeid 0x20<link>
    inet6 fe80::86fd:9db9:8d61:413e prefixlen 64 scopeid 0x20<link>
    ether 08:0c:29:dd:78:41 txqueuelen 1000 (Ethernet)
    RX packets 2462 bytes 228655 (223.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1689 bytes 207042 (202.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

2. 通过 iptables -I INPUT -p tcp --dport 22 -j LOG --log-prefix "want-in-from-port-22"; iptables -I INPUT -p tcp --dport 80 -j LOG --log-prefix "want-in-from-80" 分别配置记录 22 端口和 80 端口的情况。其中 -p 指明包的协议；--dport 指明包从哪个端口进入；-j 后面指明动作为记录进日志；-prefix 后追加记录标识以供查询。



```
root@localhost ~# iptables -I INPUT -p tcp --dport 22 -j LOG --log-prefix "want-in-from-port-22"
root@localhost ~# iptables -I INPUT -p tcp --dport 80 -j LOG --log-prefix "want-in-from-port-80"
root@localhost ~# iptables -nvL INPUT
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    0    0 LOG      tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:80
LOG flags 0 level 4 prefix "want-in-from-port-80"
    0    0 LOG      tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:22
LOG flags 0 level 4 prefix "want-in-from-port-22"
root@localhost ~#
```

3. 通过指令 iptables -t filter -I INPUT -p tcp -m tcp --dport 80 --tcp-flags

SYN,ACK,FIN,RST,URG,PSH SYN -j REJECT 拒绝 80 端口的 tcp 请求使得 curl 失败。其中主要用到了 tcp 扩展模块下的 flag，flag 后面参数分为两部分空格前表示全部要匹配的标志位，空格后表示要求为 1 的标志位。目前的配置是拒绝第一次握手。

未设置之前是可以互访的。

```

r1 PC1 PC2 PC3
64 bytes from 192.18.33.251: icmp_seq=2 ttl=64 time=0.609 ms
^C
--- 192.18.33.251 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.567/0.588/0.609/0.021 ms
[root@localhost ~]# ^C
[root@localhost ~]# service httpd start
Redirecting to /bin/systemctl start httpd.service
[root@localhost ~]# curl 192.18.33.251
MY ROUTE
Cui enbo
17069130005

r1 PC1 PC2 PC3
[root@localhost ~]# curl 192.18.33.33
outside web sever
Cui enbo
17069130005
1703011

```

设置之后不能互访，R1 长时间无响应，PC1 返回 refused。说明防火墙确实生效了。

```

[root@localhost ~]#
[root@localhost ~]# iptables -A INPUT -j REJECT
[root@localhost ~]# iptables -t filter -I INPUT -p tcp -m tcp --dport 22 --tcp-flags SYN,ACK,FIN,RST
,URG,PSH SYN -j REJECT
[root@localhost ~]# curl 192.18.33.33
^C
[root@localhost ~]# iptables -t filter -I INPUT -p tcp -m tcp --dport 80 --tcp-flags SYN,ACK,FIN,RST
,URG,PSH SYN -j REJECT
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# curl 192.18.33.33
^C
[root@localhost ~]# iptables -nvl INPUT
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
    1    60 REJECT     tcp  --  *      *        0.0.0.0/0            0.0.0.0/0            tcp dpt:80
flags:0x3F/0x02 reject-with icmp-port-unreachable
    0     0 REJECT     tcp  --  *      *        0.0.0.0/0            0.0.0.0/0            tcp dpt:22
flags:0x3F/0x02 reject-with icmp-port-unreachable
    6   397 ACCEPT     tcp  --  *      *        0.0.0.0/0            0.0.0.0/0            tcp dpt:80
    0     0 ACCEPT     tcp  --  *      *        0.0.0.0/0            0.0.0.0/0            tcp dpt:22
    0     0 LOG        tcp  --  *      *        0.0.0.0/0            0.0.0.0/0            tcp dpt:80
LOG flags 0 level 4 prefix "want-in-from-port-80"
    0     0 LOG        tcp  --  *      *        0.0.0.0/0            0.0.0.0/0            tcp dpt:22
LOG flags 0 level 4 prefix "want-in-from-port-22"
    9   540 REJECT     all  --  *      *        0.0.0.0/0            0.0.0.0/0            reject-with
icmp-port-unreachable

```

```

r1 PC1 PC2 PC3
17069130005
[root@localhost ~]# curl 192.18.33.251
curl: (7) Failed connect to 192.18.33.251:80; Connection refused

```

4. 通过 icmp 的扩展模块拒绝主机不可达报文从而使之 ping 不通。使用指令 iptables -t filter -I INPUT -p icmp --icmp-type "echo-request" -j REJECT。其中 icmp-type 后的字符串是 icmp 扩展模块的官方文档中给出的主机不可达报文的标识。

起初二者可以互相 ping 通

```
[root@localhost ~]# ping 192.18.33.33
PING 192.18.33.33 (192.18.33.33) 56(84) bytes of data.
64 bytes from 192.18.33.33: icmp_seq=1 ttl=64 time=0.398 ms
64 bytes from 192.18.33.33: icmp_seq=2 ttl=64 time=0.921 ms
64 bytes from 192.18.33.33: icmp_seq=3 ttl=64 time=0.887 ms
64 bytes from 192.18.33.33: icmp_seq=4 ttl=64 time=0.898 ms
^C
--- 192.18.33.33 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.398/0.774/0.921/0.217 ms

[root@localhost ~]# ping 192.18.33.251
PING 192.18.33.251 (192.18.33.251) 56(84) bytes of data.
64 bytes from 192.18.33.251: icmp_seq=1 ttl=64 time=0.372 ms
64 bytes from 192.18.33.251: icmp_seq=2 ttl=64 time=0.867 ms
64 bytes from 192.18.33.251: icmp_seq=3 ttl=64 time=1.08 ms
^C
--- 192.18.33.251 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2082ms
rtt min/avg/max/mdev = 0.372/0.748/1.087/0.274 ms
```

通过设置防火墙，使得 PC1 不能 ping 通 R1。

```
[root@localhost ~]# iptables -t filter -I INPUT -p icmp --icmp-type "echo-request" -j REJECT
[root@localhost ~]# ping 192.18.33.33
PING 192.18.33.33 (192.18.33.33) 56(84) bytes of data.
64 bytes from 192.18.33.33: icmp_seq=1 ttl=64 time=0.322 ms
64 bytes from 192.18.33.33: icmp_seq=2 ttl=64 time=2.32 ms
64 bytes from 192.18.33.33: icmp_seq=3 ttl=64 time=1.45 ms
64 bytes from 192.18.33.33: icmp_seq=4 ttl=64 time=0.900 ms
^C
--- 192.18.33.33 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.322/1.249/2.324/0.738 ms
```

```
[root@localhost ~]# ping 192.18.33.251
PING 192.18.33.251 (192.18.33.251) 56(84) bytes of data.
From 192.18.33.251 icmp_seq=1 Destination Port Unreachable
From 192.18.33.251 icmp_seq=2 Destination Port Unreachable
From 192.18.33.251 icmp_seq=3 Destination Port Unreachable
From 192.18.33.251 icmp_seq=4 Destination Port Unreachable
From 192.18.33.251 icmp_seq=5 Destination Port Unreachable
From 192.18.33.251 icmp_seq=6 Destination Port Unreachable
From 192.18.33.251 icmp_seq=7 Destination Port Unreachable
From 192.18.33.251 icmp_seq=8 Destination Port Unreachable
From 192.18.33.251 icmp_seq=9 Destination Port Unreachable
From 192.18.33.251 icmp_seq=10 Destination Port Unreachable
From 192.18.33.251 icmp_seq=11 Destination Port Unreachable
^C
--- 192.18.33.251 ping statistics ---
11 packets transmitted, 0 received, 100% packet loss, time 10017ms
```

5. 通过查阅资料，了解到一般情况下都是允许响应报文进入，而不允许外界主动连接主机。所以可以设置 iptables -t filter -I INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT；同时 iptables -A INPUT -j REJECT 实现。这样配置也是白名单的思想

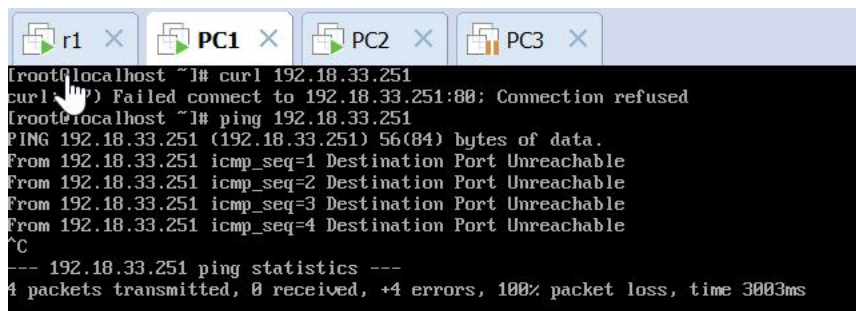
```
reject-with icmp-port-unreachable
1 60 REJECT tcp -- * * 0.0.0.0/0 0.0.0.0/0 tcp dpt:80
flags:0x3F/0x02 reject-with icmp-port-unreachable
0 0 REJECT tcp -- * * 0.0.0.0/0 0.0.0.0/0 tcp dpt:22
flags:0x3F/0x02 reject-with icmp-port-unreachable
6 397 ACCEPT tcp -- * * 0.0.0.0/0 0.0.0.0/0 tcp dpt:80
0 0 ACCEPT tcp -- * * 0.0.0.0/0 0.0.0.0/0 tcp dpt:22
0 0 LOG tcp -- * * 0.0.0.0/0 0.0.0.0/0 tcp dpt:80
LOG flags 0 level 4 prefix "want-in-from-port-80"
0 0 LOG tcp -- * * 0.0.0.0/0 0.0.0.0/0 tcp dpt:22
LOG flags 0 level 4 prefix "want-in-from-port-22"
0 0 REJECT all -- * * 0.0.0.0/0 0.0.0.0/0 reject-with
icmp-port-unreachable
[root@localhost ~]# iptables -D INPUT 2,3,4
iptables v1.4.21: Invalid rule number '2,3,4'
Try 'iptables -h' or 'iptables --help' for more information.
[root@localhost ~]# iptables -D INPUT 2 3 4
Bad argument '3'
```

```

Bad argument 3
Try 'iptables -h' or 'iptables --help' for more information.
[root@localhost ~]# iptables -D INPUT 2
[root@localhost ~]# iptables -D INPUT 3
[root@localhost ~]# iptables -D INPUT 4
[root@localhost ~]# iptables -nvL INPUT
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source                 destination            state
  0      0 ACCEPT     all  --  *      *       0.0.0.0/0              0.0.0.0/0              state RELATED,ESTABLISHED
  1     60 REJECT     tcp  --  *      *       0.0.0.0/0              0.0.0.0/0              tcp dpt:80
flags:0x3F/0x02 reject-with icmp-port-unreachable
  6    397 ACCEPT     tcp  --  *      *       0.0.0.0/0              0.0.0.0/0              tcp dpt:80
  0      0 LOG        tcp  --  *      *       0.0.0.0/0              0.0.0.0/0              tcp dpt:80
LOG flags 0 level 4 prefix "want-in-from-port-80"
  0      0 LOG        tcp  --  *      *       0.0.0.0/0              0.0.0.0/0              tcp dpt:22
LOG flags 0 level 4 prefix "want-in-from-port-22"
  0      0 REJECT     all  --  *      *       0.0.0.0/0              0.0.0.0/0              reject-with icmp-port-unreachable
[root@localhost ~]# _

```

此时 PC1 已经的 curl 和 ping 都不能再访问 R1 主机

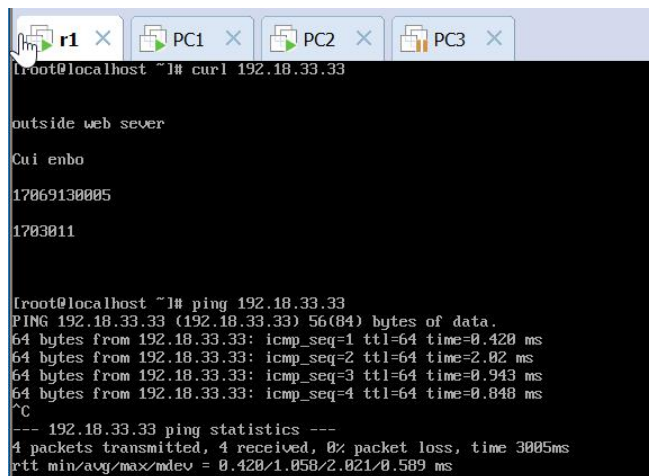


```

[root@localhost ~]# curl 192.18.33.251
curl: (7) Failed connect to 192.18.33.251:80; Connection refused
[root@localhost ~]# ping 192.18.33.251
PING 192.18.33.251 (192.18.33.251) 56(84) bytes of data.
From 192.18.33.251: icmp_seq=1 Destination Port Unreachable
From 192.18.33.251: icmp_seq=2 Destination Port Unreachable
From 192.18.33.251: icmp_seq=3 Destination Port Unreachable
From 192.18.33.251: icmp_seq=4 Destination Port Unreachable
^C
--- 192.18.33.251 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3003ms

```

但是主机 R1 却可以访问 PC1



```

[root@localhost ~]# curl 192.18.33.33
outside web sever
Cui enbo
17069130005
1703011

[root@localhost ~]# ping 192.18.33.33
PING 192.18.33.33 (192.18.33.33) 56(84) bytes of data.
64 bytes from 192.18.33.33: icmp_seq=1 ttl=64 time=0.420 ms
64 bytes from 192.18.33.33: icmp_seq=2 ttl=64 time=2.02 ms
64 bytes from 192.18.33.33: icmp_seq=3 ttl=64 time=0.943 ms
64 bytes from 192.18.33.33: icmp_seq=4 ttl=64 time=0.848 ms
^C
--- 192.18.33.33 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 0.420/1.058/2.021/0.589 ms

```

四、实验心得

通过本次实验熟悉了 iptables 的基本操作，将自己学到的一些模块指令与以前的网络知识相结合进行了一些实验。既复习了网络的内容，又对防火墙的配置加深了理解，为后面的学习打下了基础。

实验二 网络防火墙

一、实验目的

1. 通过本次实验来学习网络防火墙基本设置
2. 尝试为 iptables 中的 FORWARD 链添加规则

二、实验内容

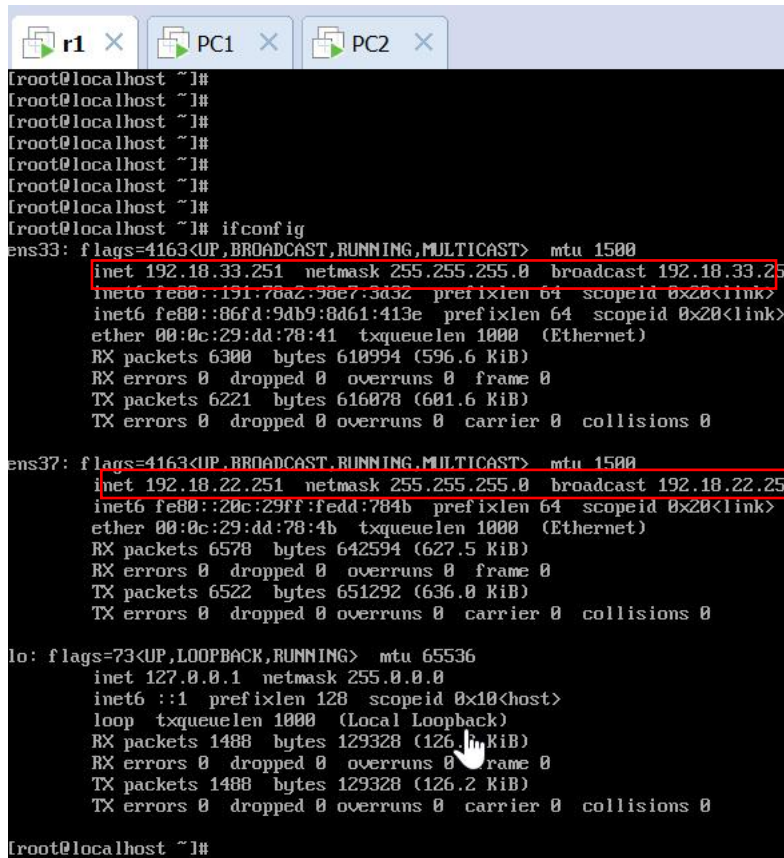
1. 我首先模拟了一个两个网段间进行通信的环境，使用三台虚机构成，一台模拟外部主机，一台充当网络防火墙，另一台充当内部主机。
2. 通过设置防火墙,阻止内外主机间通过 icmp 通信。
3. 开启内外主机 web 服务，同时打开允许发送报文，观察现象。
4. 开启允许响应报文，观察现象。

三、实验过程

1. 设置其中 PC1 的 IP 配置为 192.18.33.33 ， PC2 的 IP 配置为 192.18.22.22，虚拟机 3（R1）充当网络防火墙对 PC1 和 PC2 之间的数据包进行过滤和转发。R1 的网卡 1 的 IP 设为 192.18.33.251,网卡 2 的 IP 设为 192.18.22.251。同时将网卡 1 和 PC1 添加到同一仅主机模式的虚拟网络中，网卡 2 和 PC2 放在另一个类似的网络中。

同时通过 `route add -net 192.18.22.0/24 gw 192.18.33.251` 设置 PC1 的路由；通过 `route add -net 192.18.33.0/24 gw 192.18.22.251` 设置 PC2 的路由；

通过 `echo 1 > /proc/sys/net/ipv4/ip_forward` 来开启转发功能。



```
[root@localhost ~]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.18.33.251 netmask 255.255.255.0 broadcast 192.18.33.251
    inet6 fe80::191:78a2:98e7:3d32 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:dd:78:41 txqueuelen 1000 (Ethernet)
    RX packets 6300 bytes 610994 (596.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6221 bytes 616078 (601.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens37: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.18.22.251 netmask 255.255.255.0 broadcast 192.18.22.251
    inet6 fe80::20c:29ff:fedd:784b prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:dd:78:4b txqueuelen 1000 (Ethernet)
    RX packets 6578 bytes 642594 (627.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6522 bytes 651292 (636.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 1488 bytes 129328 (126.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1488 bytes 129328 (126.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@localhost ~]#
```

```
r1 x PC1 x PC2 x
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.18.33.33 netmask 255.255.255.0 broadcast 192.18.33.255
    inet6 fe80::191:78a2:98e7:3d32 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:2d:9a:9c txqueuelen 1000 (Ethernet)
    RX packets 40438 bytes 3559488 (3.3 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6356 bytes 614388 (599.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens37: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 00:0c:29:2d:9a:a6 txqueuelen 1000 (Ethernet)
    RX packets 24470 bytes 35350176 (33.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 5754 bytes 368083 (359.4 KiB)
```

```
r1 x PC1 x PC2 x
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.18.22.22 netmask 255.255.255.0 broadcast 192.18.22.255
    inet6 fe80::191:78a2:98e7:3d32 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:57:87:ad txqueuelen 1000 (Ethernet)
    RX packets 6553 bytes 657056 (641.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6585 bytes 639906 (624.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens37: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 00:0c:29:57:87:b7 txqueuelen 1000 (Ethernet)
    RX packets 24070 bytes 35090487 (33.4 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 5981 bytes 380694 (371.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 662 bytes 62473 (61.0 KiB)
    TX packets 662 bytes 62473 (61.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(reverse-i-search)'' :
```

2. 设置默认规则全部拒绝，使得所有数据包都不能出入。附加提示主机不可达，来确认确实是防火墙在起作用（默认为 icmp-proto-unreachable）使用命令 `Iptables -A FORWARD -j REJECT --reject-with icmp-host-unreachable`，其中-A 指明在 FORWARD 链的末尾添加功能，-j 指明对匹配到的报文拒绝并通过扩展模块返回 icmp-host-unreachable。

```

root@localhost ~#
root@localhost ~# iptables -A FORWARD -j REJECT --reject-with icmp-host-unreachable
root@localhost ~# iptables -nL FORWARD
iptables: No chain/target/match by that name.
root@localhost ~# iptables -nL FORWARD
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0      0 REJECT     all  --  *      *        0.0.0.0/0            0.0.0.0/0            reject-with
 icmp-host-unreachable

```

PC1 ping PC2

PC2 ping PC1

```

root@localhost ~# ping 192.18.22.22
PING 192.18.22.22 (192.18.22.22) 56(84) bytes of data.
From 192.18.33.251 icmp_seq=1 Destination Host Unreachable
From 192.18.33.251 icmp_seq=2 Destination Host Unreachable
From 192.18.33.251 icmp_seq=3 Destination Host Unreachable
From 192.18.33.251 icmp_seq=4 Destination Host Unreachable
From 192.18.33.251 icmp_seq=5 Destination Host Unreachable
From 192.18.33.251 icmp_seq=6 Destination Host Unreachable
From 192.18.33.251 icmp_seq=7 Destination Host Unreachable
From 192.18.33.251 icmp_seq=8 Destination Host Unreachable
From 192.18.33.251 icmp_seq=9 Destination Host Unreachable
From 192.18.33.251 icmp_seq=10 Destination Host Unreachable

root@localhost ~# ping 192.18.33.33
PING 192.18.33.33 (192.18.33.33) 56(84) bytes of data.
From 192.18.22.251 icmp_seq=1 Destination Host Unreachable
From 192.18.22.251 icmp_seq=2 Destination Host Unreachable
From 192.18.22.251 icmp_seq=3 Destination Host Unreachable
From 192.18.22.251 icmp_seq=4 Destination Host Unreachable
From 192.18.22.251 icmp_seq=5 Destination Host Unreachable
From 192.18.22.251 icmp_seq=6 Destination Host Unreachable
From 192.18.22.251 icmp_seq=7 Destination Host Unreachable
From 192.18.22.251 icmp_seq=8 Destination Host Unreachable
From 192.18.22.251 icmp_seq=9 Destination Host Unreachable
From 192.18.22.251 icmp_seq=10 Destination Host Unreachable
From 192.18.22.251 icmp_seq=11 Destination Host Unreachable
From 192.18.22.251 icmp_seq=12 Destination Host Unreachable

```

3. 尝试开启网络服务，观察到也是连接不到的。并且直接提示没有路由。表明防火墙起效

The screenshot shows two terminal windows. The left window shows the configuration of the httpd service and an attempt to connect to 192.18.22.22:80, which fails with the message 'curl: (7) Failed connect to 192.18.22.22:80; No route to host'. The right window shows the same service being started and an attempt to connect to 192.18.33.33:80, which also fails with the message 'curl: (7) Failed connect to 192.18.33.33:80; No route to host'.

4. 将 PC1 和 PC2 加入“白名单”，再次尝试。使用命令 `iptables -I FORWARD -s 192.18.22.22,192.18.33.33 -p tcp --dport 80 -j ACCEPT`。其中-s 后面指定了数据包的源地址-p 代表匹配 tcp 协议，这里省略了 -m tcp 因为模块名与协议名相同，--dport 指明端口号。由于在链尾的规则是 REJECT，所以此处设置 ACCEPT 相当于白名单的效果。

The screenshot shows a terminal window where the command `iptables -I FORWARD -s 192.18.22.22,192.18.33.33 -p tcp --dport 80 -j ACCEPT` has been entered. The terminal output shows the command being executed successfully.

观察到，依然没有反应。但是这里已经不再提示没有路由了。通过查阅资料，发现我这样设置只是允许发送报文通过防火墙，但是响应报文依然匹配到默认规则而被拦截。

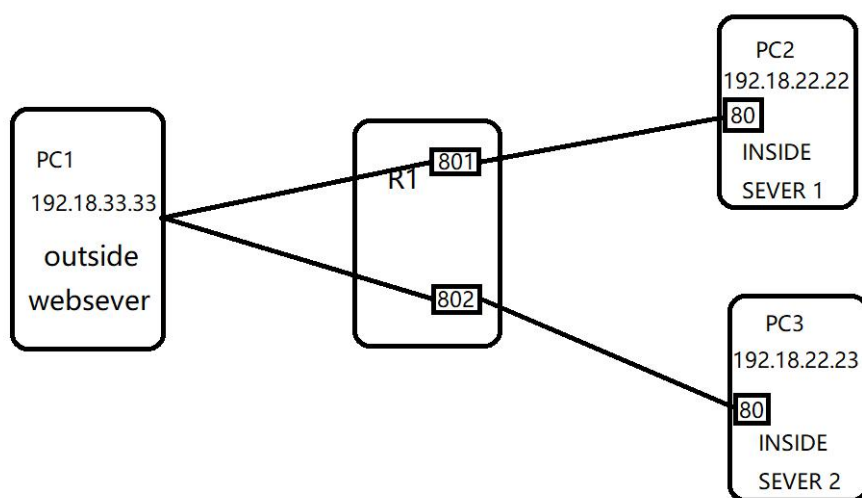
The screenshot shows a terminal window where the command `curl 192.18.22.22` has been entered. The terminal output shows the command being executed, but the result is not visible in the screenshot.

5. 设置允许响应报文通过，并观察现象。使用 `iptables -I FORWARD -d 192.18.22.22,192.18.33.33 -p tcp --sport 80 -j ACCEPT`

实验三 NAT 动作

一、 实验目的

1. 在上一个实验的基础上扩展网络防火墙功能。
2. 学习对 nat 表的修改，通过 SNAT,DNAT 动作实现网络地址转换。使得私网主机可以通过一个公网 ip 与外网主机通信。
3. 模拟公司工作环境，即内网主机可以通过公司公网 ip 访问公网，公网主机可以通过公司公网 ip 的不同端口访问不同服务器上的服务。



二、 实验内容

1. 搭建四台虚拟机构成的模拟环境，其中 PC1 代表公网主机，R1 代表公司防火墙，PC2 代表公司内部服务器 1，PC3 代表公司内部服务器 2。
2. 将 R1 添加到 PC1,PC2,PC3 的路由表中。开启 R1 的转发功能。
3. 通过设置 R1 的 iptables 的 nat 表的 POSTROUTING 链，利用 SNATT 动作使源为私网 ip 的包转换成公司的公网 ip。
4. 开启 PC1,PC2,PC3 的 httpd 服务，通过 curl 测试，观察现象。
5. 通过设置 R1 的 iptables 的 nat 表的 PREROUTING 链，利用 DNAT 动作使访问公网 801 端口的包转换成访问 PC2 的 80 端口；使得访问公网 802 端口的包转换成访问 PC3 的 80 端口。

三、 实验过程

1. 准备四台虚拟机，其中 PC1 模拟外网主机(192.18.33.33)，R1 作为网络防火墙进行 NAT 转换，PC2(192.18.22.22)和 PC3(192.18.22.23)模拟公司内的两个服务器，开放 80 端口提供服务。

同时 PC1 通过 `route add -net 192.18.22.0/24 gw 192.18.33.251`，PC2,PC3 通过 `route add -net 192.18.33.0/24 gw 192.18.22.251` 来指定 r1 作为路由。R1 通过 `echo 1 > /proc/sys/net/ipv4/ip_forward` 来开启转发功能。

```
r1 PC1 PC2
[root@localhost ~]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.18.33.33 netmask 255.255.255.0 broadcast 192.18.33.255
    inet6 fe80::191:78a2:98a7:3432 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:2d:9a:a6 txqueuelen 1000 (Ethernet)
    RX packets 466 bytes 44615 (43.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 281 bytes 46372 (45.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens37: flags=4899<UP,BROADCAST,MULTICAST> mtu 1500
    ether 00:0c:29:2d:9a:a6 txqueuelen 1000 (Ethernet)
    RX packets 147 bytes 13355 (13.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 63 bytes 5522 (5.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 16 bytes 6197 (6.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 16 bytes 6197 (6.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@localhost ~]# route add -net 192.18.22.0/24 gw 192.18.33.251
[root@localhost ~]# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.18.22.0 192.18.33.251 255.255.255.0 UG 0 0 0 ens33
192.18.33.0 0.0.0.0 255.255.255.0 U 100 0 0 ens33
[root@localhost ~]#
```

```
r1 PC1 PC2
[root@localhost ~]# route add -net 192.18.33.0/24 gw 192.18.22.251
[root@localhost ~]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.18.22.22 netmask 255.255.255.0 broadcast 192.18.22.255
    inet6 fe80::191:78a2:98a7:3432 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:57:87:ad txqueuelen 1000 (Ethernet)
    RX packets 352 bytes 55621 (54.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 308 bytes 34088 (33.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens37: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.75.142 netmask 255.255.255.0 broadcast 192.168.75.255
    inet6 fe80::20e:201:f57:07a7 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:57:87:b7 txqueuelen 1000 (Ethernet)
    RX packets 1285 bytes 91588 (89.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 556 bytes 58791 (49.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 154 bytes 18289 (17.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 154 bytes 18289 (17.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@localhost ~]# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 192.168.75.2 0.0.0.0 UG 101 0 0 ens37
192.18.22.0 0.0.0.0 255.255.255.0 U 100 0 0 ens33
192.18.33.0 192.18.22.251 255.255.255.0 UG 0 0 0 ens33
192.168.75.0 0.0.0.0 255.255.255.0 U 101 0 0 ens37
[root@localhost ~]#
```

```
r1 PC1 PC2 PC3
[root@localhost ~]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.18.22.23 netmask 255.255.255.0 broadcast 192.18.22.255
    inet6 fe80::6ea8:6915:4845:3840 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:aa:12:18 txqueuelen 1000 (Ethernet)
    RX packets 132 bytes 11163 (10.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 182 bytes 17687 (17.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 21 bytes 1870 (1.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 21 bytes 1870 (1.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@localhost ~]# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.18.22.0 0.0.0.0 255.255.255.0 U 100 0 0 ens33
192.18.33.0 192.18.22.251 255.255.255.0 UG 0 0 0 ens33
[root@localhost ~]#
```

```
r1 x PC1 x PC2 x
[root@localhost ~]#
[root@localhost ~]# echo 1 > /proc/sys/net/ipv4/ip_forward
[root@localhost ~]#
```

2. 通过 `iptables -t nat -A POSTROUTING -s 192.18.22.0/24 -j SNAT --to-source 192.18.33.251` 来指定 R1 的 NAT 转换地址。其中 `-A` 表示将 `snat` 规则添加到 `POSTROUTING` 链的末尾。其中 `POSTROUTING` 可以认为是报文发出的最后一个关卡。`-s` 用来指定报文的源地址为私网网段 `-j` 指定对匹配到的报文执行 `SNAT` 动作，转换成 `--to-source` 后的地址，即公网的 `ip` 地址。

```
r1 x PC2 x PC1 x PC3 x

[root@localhost ~]# iptables -t nat -A POSTROUTING -s 192.18.22.0/24 -j SNAT --to-source 192.18.33.251
[root@localhost ~]# iptables -t nat -nL POSTROUTING
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target prot opt in out source destination
  28 2884 POSTROUTING_direct all -- * * 0.0.0.0/0 0.0.0.0/0
  28 2884 POSTROUTING_ZONES_SOURCE all -- * * 0.0.0.0/0 0.0.0.0/0
  28 0 SNAT all -- * * 192.18.22.0/24 0.0.0.0/0 to:192.18.33.251
  0 0 SNAT all -- * * 192.18.22.0/24 0.0.0.0/0 to:192.18.33.251
```

3. 开启 PC1 和 PC2 的 web 服务。同时为了便于观察，通过 `iptables -I INPUT -p tcp --dport 80 -m state --state NEW -j LOG --log-prefix "want-in-from-22"` 指令将 80 端口收到的新建连接的包记录到日志中，同时标识为“want-in-from-22”（来自 192.18.22.xx 的包）。

```
r1 x PC1 x PC2 x PC3 x

[root@localhost ~]# iptables -I INPUT -p tcp --dport 80 -m state --state NEW -j LOG --log-prefix "want-in-from-22"
[root@localhost ~]# iptables -nL INPUT
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target prot opt in out source destination
  0 0 LOG tcp -- * * 0.0.0.0/0 0.0.0.0/0 tcp dpt:80
state NEW LOG flags 0 level 4 prefix "want-in-from-22"
```

4. 在 PC2 端使用 `curl 192.18.33.33` 访问公网主机。同时通过 `tail -f /var/log/messages` 来查看防火墙日志。

PC2 成功访问到 PC1

```
r1 x PC1 x PC2 x PC3 x

[root@localhost ~]# curl 192.18.33.33

outside web sever
Cui enbo
17869138805
1783811
```

这里源地址已经转换成公司的公网 `ip`，表明 `SNAT` 动作生效。完成了实验设计的第一部分，私网主机通过 `SNAT` 转换成公网 `ip` 访问公网服务器。


```
r1 x PC1 x PC2 x PC3 x
=49368 DPT=80 WINDOW=29200 RES=0x00 SYN URGF=0
Nov 25 01:53:39 localhost kernel: want-in-from-22IN=ens33 OUT= MAC=00:0c:29:2d:9a:9c:00:0c:29:dd:78:
41:08:00 SRC=192.18.22.22 DST=192.18.33.33 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=4239 DF PROTO=TCP SPT
=49370 DPT=80 WINDOW=29200 RES=0x00 SYN URGF=0
Nov 25 01:54:09 localhost kernel: want-in-from-22IN=ens33 OUT= MAC=00:0c:29:2d:9a:9c:00:0c:29:dd:78:
41:08:00 SRC=192.18.22.22 DST=192.18.33.33 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=20246 DF PROTO=TCP SP
T=49372 DPT=80 WINDOW=29200 RES=0x00 SYN URGF=0
Nov 25 01:54:17 localhost kernel: want-in-from-22IN=ens33 OUT= MAC=00:0c:29:2d:9a:9c:00:0c:29:dd:78:
41:08:00 SRC=192.18.22.22 DST=192.18.33.33 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=49557 DF PROTO=TCP SP
T=49374 DPT=80 WINDOW=29200 RES=0x00 SYN URGF=0
Nov 25 01:54:40 localhost kernel: want-in-from-22IN=ens33 OUT= MAC=00:0c:29:2d:9a:9c:00:0c:29:dd:78:
41:08:00 SRC=192.18.33.251 DST=192.18.33.33 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=59977 DF PROTO=TCP S
PT=49378 DPT=80 WINDOW=29200 RES=0x00 SYN URGF=0
```

5. 接下来通过设置 DNAT 动作使得公网主机可以通过公网 ip 的不同端口访问私网内的不同服务器。首先通过 iptables -t nat -I PREROUTING -d 192.18.33.251 -p tcp --dport 801 -j DNAT --to-destination 192.18.22.22:80 设置。使得访问公网 801 端口的包被转发到私网内 192.18.22.22 的 80 端口。其中 -d 匹配目的地址为公司公网 ip 的包。通过 --dport 指定进入端口。-j 指定动作为 DNAT 将这个包转发到 --to-destination 对应的地址。

```
r1 x PC1 x PC2 x PC3 x
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# iptables -t nat -I PREROUTING -d 192.18.33.251 -p tcp --dport 801 -j DNAT --to-d
estination 192.18.22.22:80
[root@localhost ~]# iptables -t nat -I PREROUTING -d 192.18.33.251 -p tcp --dport 801 -j DNAT --to-d
estination 192.18.22.22:80
[root@localhost ~]# iptables -t nat -nVL PREROUTING
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source        destination    tcp dpt:801
0 0 DNAT        tcp -- *      *       0.0.0.0/0     192.18.33.251
to:192.18.22.22:80
0 0 DNAT        tcp -- *      *       0.0.0.0/0     192.18.33.251
to:192.18.22.22:80
0 0 DNAT        tcp -- *      *       0.0.0.0/0     192.18.33.251
to:192.18.22.23:80
0 0 DNAT        tcp -- *      *       0.0.0.0/0     192.18.33.251
to:192.18.22.22:80
9 540 DNAT      tcp -- *      *       0.0.0.0/0     192.18.33.251
to:192.18.22.22:80
10 600 DNAT      tcp -- *      *       0.0.0.0/0     192.18.33.251
to:192.18.22.23:80
1 60 DNAT       tcp -- *      *       0.0.0.0/0     192.18.33.251
to:192.18.22.23:80
2 120 DNAT      tcp -- *      *       0.0.0.0/0     192.18.33.251
to:192.18.22.22:80
2 120 DNAT      tcp -- *      *       0.0.0.0/0     192.18.33.251
9 to:192.18.22.22:3389
225 28575 PREROUTING_direct all -- *      *       0.0.0.0/0     0.0.0.0/0
225 28575 PREROUTING_ZONES_SOURCE all -- *      *       0.0.0.0/0     0.0.0.0/0
225 28575 PREROUTING_ZONES all -- *      *       0.0.0.0/0     0.0.0.0/0
```

6. 通过 PC1 来分别访问公网 ip 的 801 端口和 802 端口，观察发现访问成功，说明 DNAT 动作生效。

```
r1 x PC1 x PC2 x PC3 x
[root@localhost ~]#
[root@localhost ~]# curl 192.18.33.251:801
inside sever 1

Cui enbo
17069130005
[root@localhost ~]# curl 192.18.33.251:802
inside sever 2

Cui enbo
17069130005
```

四、实验心得

通过本次实验实验，完成了对 **PREROUTING** 和 **POSTROUTING** 两条链的配置，学习了关于 **NAT** 的相关知识。模拟环境的过程中经常遇到问题，通过不断的解决这些随机的的问题，大大提高了独立分析和解决问题的能力。

通过查阅资料，后来发现在实际中为了适应 **IP** 的动态变换，经常使用 **MASQUERADE** 动作，即 `iptables -t nat -I POSTROUTING -s 192.18.0.0/16 -o ens33 -j MASQUERADE`，这样无需指明 NAT 转换的目标地址，可以适应动态变化。此外，还了解到 **PREROUTING** 链还有一个动作是 **REDIRECT** 重定向，可以将本机端口进行映射，如 `iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 801`，可以将要进入 80 端口的，转到 801 端口。

实验四 TCP 连接与 Dos 防御

一、实验目的

1. 通过 socket 编程实现 TCP 连接。
2. 通过设置防火墙，限制每个 IP 只能建立两个链接。

二、实验内容

1. 流式套接字提供可靠，面向链接的服务，基于 TCP 协议。所以本次实验主要使用流式套接字，以及相关函数：socket 来建立链接，bind 来监听端口，accept 用于服务器端接收链接，recv 用于服务器端接收数据，send 用于客户端发送数据。通过以上函数实现 TCP 连接。
2. 通过设置路由器的防火墙来防御实验一中，通过拒绝 TCP 第一次握手可能导致的拒绝服务攻击。
3. 通过两台虚拟机进行测试。分别为 r1(192.168.157.1) 和 Outer Host(192.168.157.2)

三、实验过程

1. 编写服务器端代码

```
#define SERVER_PORT 803
int main()
{
    int serverSocket;
    struct sockaddr_in server_addr;
    struct sockaddr_in client_addr;           //两个套接字结构体变量，分别表
                                              //示客户端和服务端

    int addr_len = sizeof(client_addr);
    int client;
    char buffer[200];
    int iDataNum;
    if((serverSocket = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    { //通过 socket() 建立连接，失败时会返回-1
        perror(" socket" );
        return 1;
    }

    bzero(&server_addr, sizeof(server_addr)); //初始化服务器端套接字

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(SERVER_PORT);
    //将 IP 设为本机 IP 地址，本实验中为服务器 192.168.157.1
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    if(bind(serverSocket, (struct sockaddr *)&server_addr, sizeof(server_addr))
< 0)
```

```

{
    perror(" connect" );
    return 1;
}

if(listen(serverSocket, 5) < 0)           //将服务器设置为监听状态
{
    perror(" listen" );
    return 1;
}

while(1)
{
    //循环监听 803 端口的消息
    //调用 accept 函数后，会进入阻塞状态
    //accept 返回一个套接字的文件描述符，这样服务器端便有两个套
    //接字的文件描述符，
    //serverSocket 和 client。
    //serverSocket 仍然继续在监听状态，client 则负责接收和发送数据
    //clientAddr 是一个传出参数，accept 返回时，传出客户端的地址
    //和端口号
    //addr_len 是一个传入-传出参数，传入的是调用者提供的缓冲区的
    //clientAddr 的长度，以避免缓冲区溢出。
    //传出的是客户端地址结构体的实际长度。
    //出错返回-1
    printf(" Listening on port: %d\n", SERVER_PORT);
    client = accept(serverSocket, (struct sockaddr*)&client_addr,
(socklen_t*)&addr_len);
    if(client < 0)
    {
        perror(" accept" );
        continue;
    }
    printf(" \nrecv client data...\n" );
    printf(" IP is %s\n", inet_ntoa(client_addr.sin_addr));
    printf(" Port is %d\n", htons(client_addr.sin_port));
    while(1)
    {
        iDataNum = recv(client, buffer, 1024, 0);
        if(iDataNum < 0)
        {
            perror(" recv" );
            continue;
        }
    }
}

```



```

        buffer[iDataNum] = '\0';
        if(strcmp(buffer, " quit" ) == 0)
            break;
        printf(" %drecv data is %s\n" , iDataNum, buffer);
        send(client, buffer, iDataNum, 0);
    }
}
return 0;
}

```

2. 编写客户端程序

```

#define SERVER_PORT 803
int main()
{

    int clientSocket;

    struct sockaddr_in server_addr;        //初始化服务器端套接字
    char sendbuf[200];
    char recvbuf[200];
    int iDataNum;
    if((clientSocket = socket(AF_INET, SOCK_STREAM, 0)) < 0)    //建立连接
    {
        perror(" socket" );
        return 1;
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(SERVER_PORT);

    server_addr.sin_addr.s_addr = inet_addr(" 192.168.157.1" );
    //将服务器地址设置为 192.168.157.1

    if(connect(clientSocket,      (struct      sockaddr      *)&server_addr,
sizeof(server_addr)) < 0)
    {
        perror(" connect" );
        return 1;
    }

    printf(" connect with destination host...\n" );

    while(1)
    {
        //循环等待用户输入，并通过 send 函数发送给服务器
    }
}

```

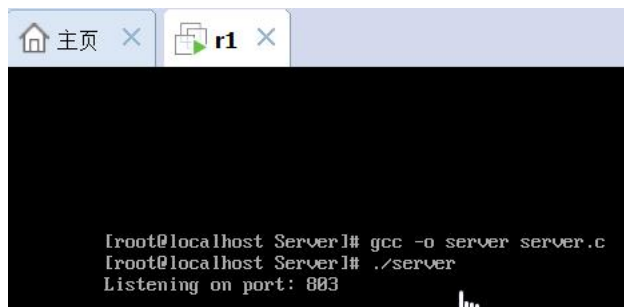
```

printf(" Input your world:>" );
scanf(" %s" , sendbuf);
printf(" \n" );

send(clientSocket, sendbuf, strlen(sendbuf), 0);
if(strcmp(sendbuf, " quit" ) == 0)
    break;
iDataNum = recv(clientSocket, recvbuf, 200, 0);
recvbuf[iDataNum] = '\0';
printf(" recv data of my world is: %s\n" , recvbuf);
}
close(clientSocket);
return 0;
}

```

3. 先编译运行服务器端，再编译运行客户端。并通过客户端向服务器发数据。

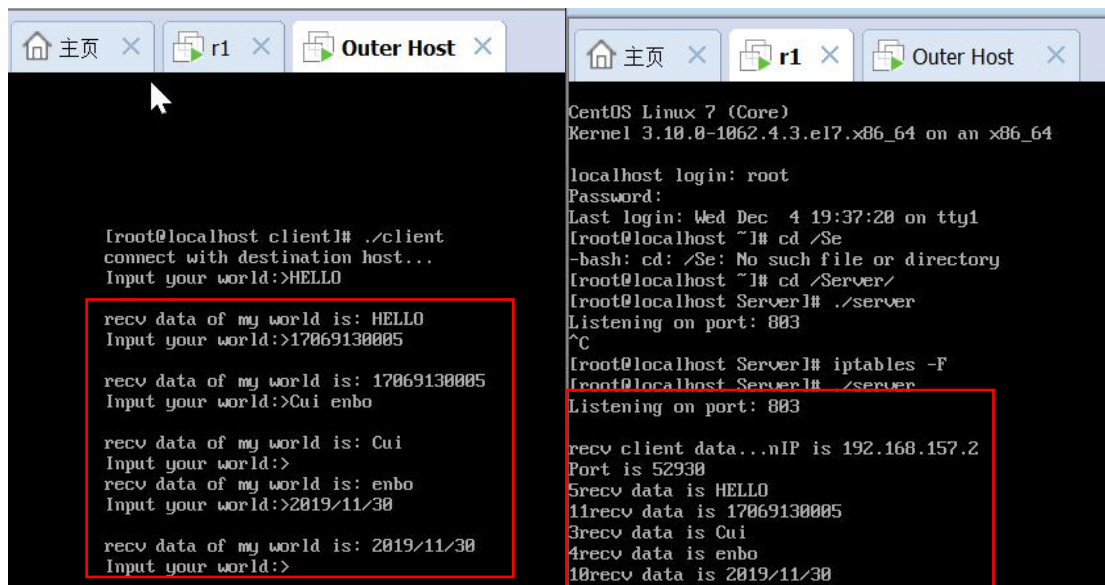


```

[root@localhost Server]# gcc -o server server.c
[root@localhost Server]# ./server
Listening on port: 803

```

发现确实实现了连接，能通过 803 端口传输数据。



```

[root@localhost client]# ./client
connect with destination host...
Input your world:>HELLO

recv data of my world is: HELLO
Input your world:>17069130005

recv data of my world is: 17069130005
Input your world:>Cui enbo

recv data of my world is: Cui
Input your world:>
recv data of my world is: enbo
Input your world:>2019/11/30

recv data of my world is: 2019/11/30
Input your world:>

```

```

CentOS Linux 7 (Core)
Kernel 3.10.0-1062.4.3.el7.x86_64 on an x86_64


localhost login: root
Password:
Last login: Wed Dec 4 19:37:20 on tty1
[root@localhost ~]# cd /Se
-bash: cd: /Se: No such file or directory
[root@localhost ~]# cd /Server/
[root@localhost Server]# ./server
Listening on port: 803
^C
[root@localhost Server]# iptables -F
[root@localhost Server]# ./server
Listening on port: 803

recv client data...nIP is 192.168.157.2
Port is 52930
5recv data is HELLO
11recv data is 17069130005
3recv data is Cui
4recv data is enbo
10recv data is 2019/11/30

```

4. 与实验一中拒绝 TCP 第二次握手相结合 iptables -t filter -I INPUT -p tcp -m tcp --tcp-flags SYN,ACK,FIN,RST,URG,PSH SYN,ACK -j REJECT。模拟 Dos 攻击的防御。

5. 通过设置 r1 的防火墙限制每个 IP 至多与 803 端口建立两个连接。使用命令 `iptables -I INPUT -p tcp --dport 803 -m connlimit --connlimit-above 2 -j REJECT`。其中 `connlimit` 扩展模块用于限制每个 IP 地址同时连接到 server 的数量。此扩展模块在不指定 IP 的时候，会默认针对每个用户端 IP，通过 `above` 选项限制匹配连接数量超过 2 的 IP，并对这个 IP 后续的包执行 `REJECT` 操作。

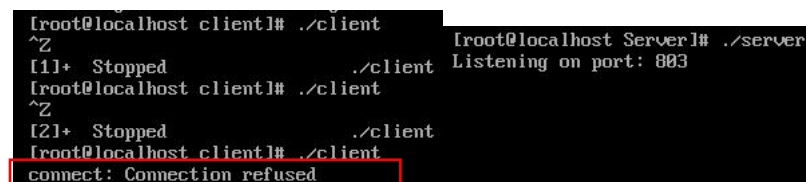


```

^C
[root@localhost Server]# iptables -F
[root@localhost Server]# iptables -I INPUT -p tcp --dport 803 -m connlimit --connlimit-above 2 -j REJECT

```

6. 尝试建立三个 TCP 连接。



```

[root@localhost client]# ./client
^Z
[1]+  Stopped                  ./client
[root@localhost client]# ./client
^Z
[2]+  Stopped                  ./client
[root@localhost client]# ./client
connect: Connection refused

```

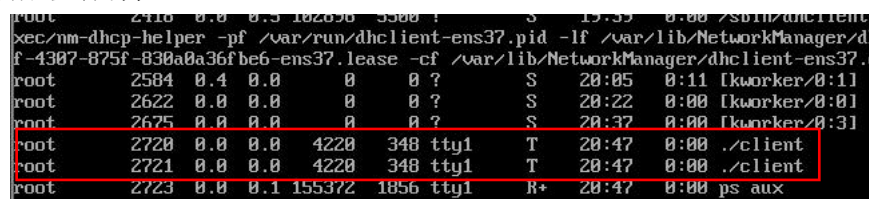
```

[root@localhost Server]# ./server
Listening on port: 803

```

第三个链接惨遭拒绝。服务器端没有任何反应

7. 查看进程列表确认进程数量。发现确实只能运行两个客户端程序。表面防火墙配置成功。



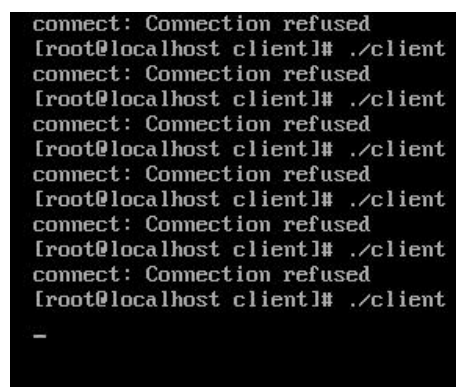
```

root      2418  0.0  0.3 102896 3500 ?        S    19:35   0:00 /sbin/dhclient
xec/nm-dhcp-helper -pf /var/run/dhclient-ens37.pid -lf /var/lib/NetworkManager/dhclient-ens37.conf -4307-875f-830a0a36fbe6-ens37.lease -cf /var/lib/NetworkManager/dhclient-ens37.conf
root      2584  0.4  0.0      0  0 ?        S    20:05   0:11 [kworker/0:1]
root      2622  0.0  0.0      0  0 ?        S    20:22   0:00 [kworker/0:0]
root      2625  0.0  0.0      0  0 ?        S    20:32   0:00 [kworker/0:3]
root      2720  0.0  0.0  4220   348 tty1    T    20:47   0:00 ./client
root      2721  0.0  0.0  4220   348 tty1    T    20:47   0:00 ./client
root      2723  0.0  0.1 155372 1856 tty1    R+   20:47   0:00 ps aux

```

8. 此外，发现即使将客户端两个进程杀掉后也不能建立连接。我认为是因为我拒绝了 TCP 第二次握手，所以在服务器那端仍然是两条连接的状态。

在等待约一分半之后，又能连接了，说明 PC1 的 Dos 攻击也成功了



```

connect: Connection refused
[root@localhost client]# ./client
connect: Connection refused
[root@localhost client]# ./client
connect: Connection refused
[root@localhost client]# ./client
connect: Connection refused
[root@localhost client]# ./client
connect: Connection refused
[root@localhost client]# ./client
connect: Connection refused
[root@localhost client]# ./client
-

```

四、实验心得

在刚开始学习 `iptables` 的时候，就想模拟防御 dos 攻击，但是一直没找好测

试设置的方法。本次实验通过 `socket` 编程实现了建立长时间的 TCP 链接，完成了规则的测试，感觉整个人都圆满了。

总的来说，本次大作业涉及的方面比较广，除了本学期的安全相关知识，还有计算机网络，虚拟机等各方面应知应会的知识。借着本次实验的机会，学习，复习，令我受益匪浅。