iptables防火墙基础

- 前言：

这是我前面那么多节学习以来最难的一节了，iptables也是Linux里一个特别难啃的骨头，涉及了大量网络知识。我其实学的一知半解，所以估计用自己的话来简单概括一些抽象的知识点应该是很难做到的了。恩抄就完事了。言归正传，iptables是系统自带的，工作于系统的底层（内核层），不像Windows上那些xx管家啊，都是基于应用层的傻瓜式使用。

- 正文：

1、iptables的作用

Linux中的防火墙，由ipfirewall发展到ipchains再发展到iptables而来。表面上是一个命令（iptables直接使用），也是一个服务（systemctl使用），可以用-h查看参数手册，但它的使用十分复杂，需要掌握大量防火墙的知识。CentOS 7.x下不带iptables软件，需要用yum来安装，yum -y install iptables-services。

```
Install  1 Package

Total download size: 52 k
Installed size: 23 k
Downloading packages:
iptables-services-1.4.21-35.el7.x86_64.rpm                                  |  52 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : iptables-services-1.4.21-35.el7.x86_64                                       1/1
  Verifying  : iptables-services-1.4.21-35.el7.x86_64                                       1/1

Installed:
  iptables-services.x86_64 0:1.4.21-35.el7

Complete!
[root@localhost ~]# systemctl status iptables
● iptables.service - IPv4 firewall with iptables
   Loaded: loaded (/usr/lib/systemd/system/iptables.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
[root@localhost ~]# systemctl start iptables
[root@localhost ~]# systemctl status iptables
● iptables.service - IPv4 firewall with iptables
   Loaded: loaded (/usr/lib/systemd/system/iptables.service; disabled; vendor preset: disabled)
   Active: active (exited) since Mon 2021-09-20 10:33:37 CST; 1s ago
  Process: 1489 ExecStart=/usr/libexec/iptables/iptables.init start (code=exited, status=0/SUCCESS)
 Main PID: 1489 (code=exited, status=0/SUCCESS)

Sep 20 10:33:37 localhost.localdomain systemd[1]: Starting IPv4 firewall with iptables...
Sep 20 10:33:37 localhost.localdomain iptables.init[1489]: iptables: Applying firewall rules: [... ]
Sep 20 10:33:37 localhost.localdomain systemd[1]: Started IPv4 firewall with iptables.
Hint: Some lines were ellipsized, use -l to show in full.
[root@localhost ~]# ps -ef | grep iptables
root      1540  1383  0 10:34 tty1     00:00:00 grep --color=auto iptables
[root@localhost ~]# netstat -tnlp | grep iptables
[root@localhost ~]# _
```

安装好后，通过systemctl开启iptables服务。通过ps查看进程和netstat查看网络连接信息，都看不到iptables，这不是个防火墙服务吗，怎么哪都看不到?

因为防火墙真正工作的，是内核中一个叫做Netfilter的功能框架，Linux内核利用其自身对于网络数据包的流向判断间接起到防火墙功能。而iptables只是一个桥梁，帮助我们使用内核的框架，可以理解为iptables是使用这个防火墙的工具，一个专有命令行，在外围里。防火墙在内核里，而不是一个软件跑在系统上，故看不到它的。但习惯上还是管Linux的防火墙叫iptables。

2、4个表

iptables在使用过程中，有四种标准化预定义的表（tables），每一种表代表了一种使用的类型或者说方向。分为：filter、nat、mangle、raw，前两种必掌握，出场率极高，后两种不常用，属于高级拓展。

①filter。最熟悉的表，大致上就是我们平时在Windows上各种安全管家使用的，设置一道关卡，严查外来的东西安不安全、符不符合规则，OK就放行，NO就丢进垃圾桶。但是Linux的filter是很严格的很彻底的，和Windows的随意不一样。

```
[root@localhost ~]# iptables -A INPUT -s 172.16.0.0/16 -j DROP
[root@localhost ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere             state NEW tcp dpt:ssh
REJECT     all  --  anywhere             anywhere             reject-with icmp-host-prohibited
DROP       all  --  172.16.0.0/16        anywhere
```

第一条命令是指加上一条规则，它对于进入系统的且来源是172.16.0.0/16这个范围的网络请求全部抛弃（DROP，与之对应的是ACCEPT）。第二条加上 -L 参数即可查看已有的规则，可以看到成功添加了。这个是最简单的最傻瓜的防火墙配置了。

②nat。和过滤不再有关系，全称是Network Address Translation（网络地址转换），是一种对数据包的特殊处理。其主要通过iptables把原本发过来的请求更改其原始的目的地址或者其来源地址（port端口），以达到请求转发的目的。例如，我要发送请求到服务器IP地址A，结果iptables处理后目的地址变成了服务器IP地址B，nat把这个请求转交了；或者说，本来请求发到一个服务器IP地址C上的端口80中，结果nat改变它的目的地址，转发到了8080端口的应用处。至于为什么喜欢这么干，我不到啊。它的命令设置很复杂，后面学了相关知识再实践。

③mangle。第二优先级，在数据包经过路由表之前根据规则修改数据包的一些标志位，以便二次处理。比较常用的是策略路由，就是上网分路线，例如内网要正常上网就连联通，上外网就连电信，前提是Linux做上网的网关并且连接两条ISP公网路线。传统的修改静态路由是根据IP来定制的，而上述的策略路由是偏向服务类的路由的，因此这个策略路由不好按传统方式做，mangle就派上用场了。

```
#在经过路由表前按照需求给某一类请求打上标签
iptables -t mangle -A PREROUTING -i eth0 -p tcp --dport 80:443 -j MARK --set-mark
1
iptables -t mangle -A PREROUTING -i eth0 -p tcp --dport 80:443 -j MARK --set-mark
1
#给不同目的端口打上标签1和2后，要定制两张路由表10和20，并且对两种标签分别做静态路由
ip route add default via 202.106.x.x dev eth1 table 10
ip route add default via 211.108.x.x dev eth2 table 20
ip rule add from all??fwmark 1 table 10
ip rule add from all??fwmark 2 table 20
```

④raw。第一优先级，针对一个数据包，让它可以跳过连接跟踪和NAT。Linux中有一个叫做ip_conntrack的模块，专门用于对所有进入内核的数据包进行状态标记（后面会说）和跟踪，并存在一张表里。如果设置了raw规则，那么数据包不再被ip_conntrack追踪记录，并且还能跳过nat规则。使用动作是 NOTRACK 。

3、5个链

还记得前面的ipchains吗，就是这个了。表的概念只是一个逻辑大分类，把制定的规则按照不同的功能进行划分，真正的制定规则就要用到这5个链了。可以理解为选了表确定努力方向后，下一步进行功能的选择（链）。最后加点修饰，一条规则就成了。

默认5条链，这5条链对应着一个数据包流向的每个步骤

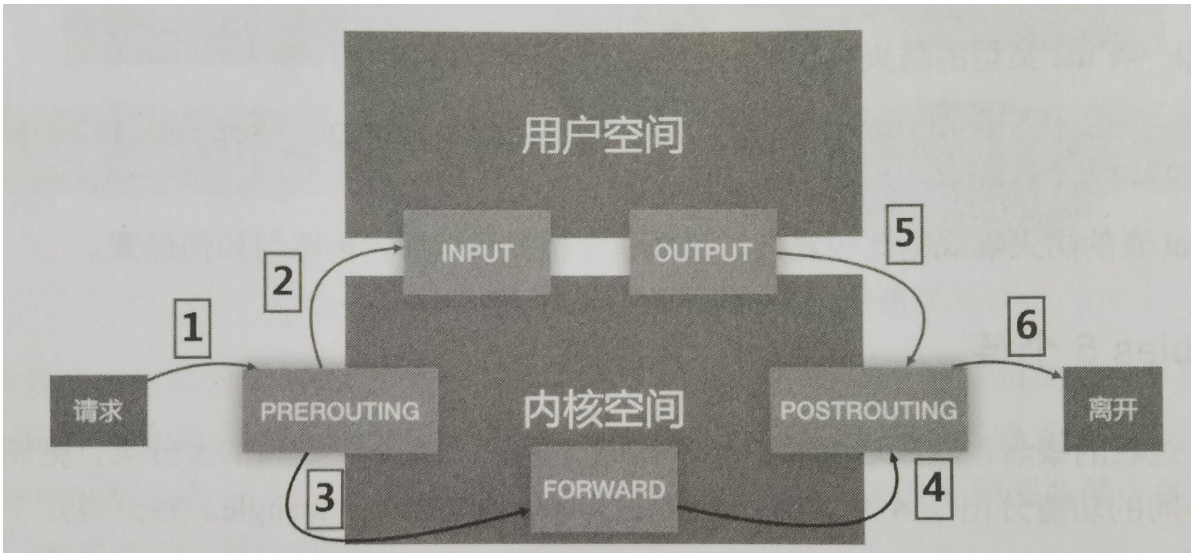


图 16.10　用户空间和内核空间

①PREROUTING：数据包进入路由表之前（请求要先按照一定的规矩确定后续方向）

②INPUT：通过路由表后，目的是本地（外来信息进入我的机子）

③FORWARD：通过路由表后，目的不是本地（要转发了，就不留下来了）

④OUTPUT：由本机产生，向外发送（我的机子要向外发信息）

⑤POSTROUTING：发送到网卡接口之前（发到网卡就意味着完成和其他机子的信息交换了）

任何种类的网络请求都是以上流程，比较复杂，后面我们细细道来，希望真的能细细道来，哭。

4、链和内核功能块的关系

先理解几个名词：

①内核空间：数据包刚到达时第一个进入的处理场所，指的是系统底层，也就是底层对网络数据包的判断、修改、转发、路由等功能

②用户空间：用户肉眼可见的、可以操作的，例如什么图形界面啊、服务软件啊一类的，除去内核外的部分都是了。数据包刚进系统时，不管有没有设置iptables，都要经过内核的筛选，OK的才能到用户空间。

③路由功能：Linux的内核对数据包的转发功能。自己作为路由器（FORWARD路由转发），将数据包发送到其他网络或者非本机地址。这是直连路由的。注意，内核有个FORWARD路由功能，而链里有个FORWARD防火墙链，这两个直接相关。

五个链其实有三层含义：

①iptables命令直接调用某一个链，启用某个新规格。iptables -t 表名字 -A 链名字；

②iptables这配置的5个链，往深处走对应内核netfilter框架的5个功能模块；

③5个功能模块再往下对应内核代码中的5个内核钩子函数，真正底层做事的是这些钩子函数对数据包的处理判断。

```
[root@localhost ~]# systemctl stop iptables
[root@localhost ~]# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@localhost ~]# systemctl start iptables
[root@localhost ~]# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:22
REJECT     all  --  0.0.0.0/0            0.0.0.0/0           reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
REJECT     all  --  0.0.0.0/0            0.0.0.0/0           reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@localhost ~]# _
```

如图，记住iptables始终是命令和服务而已，即使关掉了，但是内核中的框架和钩子代码都是存在的，”关“——其实是内核防火墙下的所有链规则全设成通过，就好像防火墙不存在一样。然而内核防火墙作为内核处理数据包的一部分，永远都关不掉。

5、5个链的具体功能

时刻要记住这张图：

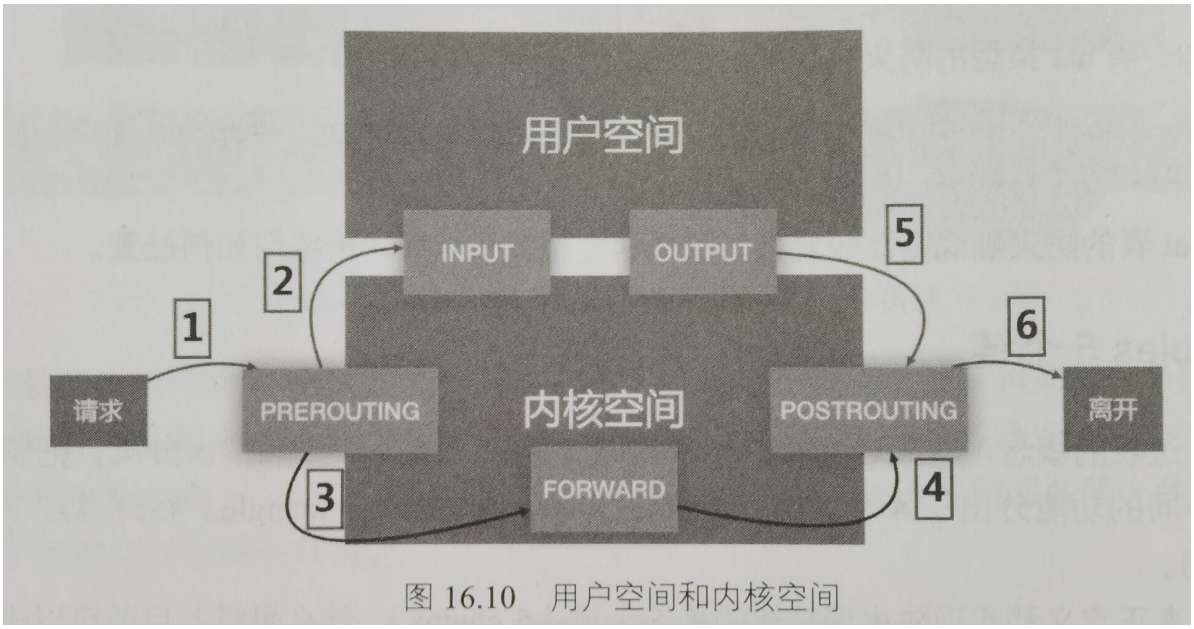


图 16.10　用户空间和内核空间

①PREROUTING：

字面意思，路由前，在数据包经过路由表对数据包进行第一次判断，结果只有两种：需要进入路由转发（到别的IP去）；不需要路由转发（到本机IP上）。PREROUTING是第一道关卡，它判断数据包的目的地址、端口。可能有人会问（这也是我最开始的疑问），一个数据包发到这个机子上，不就是访问这个机子的IP吗，怎么还发给别的地址去了。可谁让人家设计就这样的呢，PREROUTING链和POSTROUTING链的本质就是修改目的地址/端口和源地址/端口，也就是这样Linux才能做一个路由器。但是这种转发修改地址要自己定制规则的，例如一个数据包到PREROUTING时，它会根据事先制定的规则来判断是否要修改地址。

```
[root@localhost ~]# iptables -t nat -A PREROUTING -s 172.16.0.0/8 -d 192.168.56.102 -p tcp -j DNAT -
-to-destination 192.168.56.103
[root@localhost ~]# iptables -t nat -L -n
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       tcp  --  172.0.0.0/8          192.168.56.102       to:192.168.56.103

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
[root@localhost ~]#
```

-t 后接 nat 表，-A 后接链名，-s后接请求的源地址，-d后接请求的目的地址，-p后接请求的传输协议，-j后接对请求的动作（DNAT后面那一串指改变目标地址到192.168.56.103），整个流程下来就是改变destination 192.168.56.102 to 192.168.56.103。一个请求符合这条规则，就按这个规则行事改变目的地址。

修改了目的地址之后呢，不符合规则不修改之后呢？PREROUTING链的判定分为两个步骤：第一个是是否满足这条规则进行地址修改；第二个就是根据结果决定走上路还是下路，上路就是过滤型防火墙，下路就是nat型防火墙。

PREROUTING上路判定：不符合规则，不修改目的地址，数据包往上走，被允许传入用户空间，即PREROUTING——INPUT。可以认为，网络请求在内核空间中是无害的，真正的破坏力在于数据包进入高层应用后对高层应用的修改。走上路，相当于走到了过滤型防火墙（filter表），里面可以配置的链有三个：INPUT、OUTPUT、FORWARD。

PREROUTING下路判定：符合规则，修改目的地址，数据包走下路，要在内核空间里进行转发到其他非本机地址上。之后交给FORWARD链。注意，我们之前也讲过，Linux内核空间原本就有FORWARD的功能（转发），我们在上路判定也说过之后可以配置FORWARD链，那是netfilter里的自带的一个FORWARD链，和其他4个链一样做数据包判断用的。PREROUTING——FORWARD——路由表——路由功能判定——POSTROUTING——下一层网络目的地，这是下路的完整的数据包流程。

##路由功能判定，结果可能是直连路由（同网段）、路由（不同网段，Linux的路由功能）、丢弃（符合设定的禁止规则，不转发）。

②POSTROUTING：

这是判定为下路时用到的链，其作用是决定数据包是否需要改变来源地址。这个就比较简单了，没什么好讲的，因此这部分我们顺便深入理解PREROUTING/POSTROUTING这一对玩意儿。

##题外话1，先理解以下三个概念：

1）路由表：一张可见可更改的表，当数据包发到Linux时，系统（或内核）根据这张表中定义好的信息决定这个包的后续流程。官方定义是，路由器或其他设备上存储的一张路由信息表，内含到达特定网络终端的路径，某些情况下还有与这些路径相关的度量。

2）路由功能和路由器：路由功能就是具有根据路由表信息决定数据包去向，跨网段时则必须要网关设备，即路由器。网关设备连接两个不同网段，可以说一个硬件路由器（平时家用的小型路由器），也可以是开启了路由功能的Linux主机。

3）转发：对一个数据包进行的特殊处理，可以理解为传输过程中的一种中转或者衔接。下图为开启关闭转发功能的指令：

```
[root@localhost ~]# echo 0 > /proc/sys/net/ipv4/ip_forward
[root@localhost ~]# echo 1 > /proc/sys/net/ipv4/ip_forward
[root@localhost ~]#
```

开启转发功能后，Linux自身形成了一个小路由器，可以当网关来使用；也只有开了转发功能，iptables的nat表才能使用（数据修改NAT）。

明白后，可以理解为什么PREROUTING只能修改目标地址，而POSTROUTING只能修改源地址了。在查路由表时，就能询问到去目标地址的路，所以在查路由表前的处理（PREROUTING）必须提早改目标地址，否则进路由表后查的还是原来的目标地址；查完路由表并经过判定后才改源地址，是为了实现SNAT技术，也可以理解为如果早修改源地址了，到时候数据包回来查记录，查不到最开始的发出地址，这样就跑到了更改后并登记的源地址上了。

##题外话2，DNAT和SNAT技术：

DNAT（Destination Network Address Translation），目标地址转换。有很多用处，如公司内部的某个站点使用DNAT对外开放，但不能让员工知道这个站点，员工不能直接登录这个站点，但可以通过连接公司提供的一个地址，然后再登录机器上改变目标地址连到内部站点上，这种应用在堡垒机（跳板机）的实现中增强了安全系数，用户先登录跳板机，才能连到其他后端机器（间接登录），可以实现用户行为追踪保护后端登录安全；另外，可以进行服务器本机端口转换，把要用的默认端口禁止掉，要用其他端口DNAT改变目标端口连到这些默认端口。

SNAT（Source Network Address Translation），源地址转换。这个更好理解了，就是把内网转变成公网（数据包源地址改成公网IP），这样才能上网冲浪，上一节的网络基础讲过。毕竟内网IP不能上网的嘛，所以在POSTROUTING链里把源地址从本机内网网卡改成本机公网网卡的地址，在防火墙处留下一条记录，然后送到互联网对端，之后返回时回到Linux网关，通过之前的记录发回给内网IP的对应机子上。

##题外话3，命令行

先熟悉以下参数：

-t 指定在哪张表下设置规则。

-A 指定一个链并指定本条规则是追加到链末尾，对应的 -I 将规则放在链前

-s 限定源地址的符合规则

-d 限定目标地址的符合规则

-j 执行动作和处理

-p 指定协议

实践，纯手打：

```
#PREROUTING实现目标地址全转发
iptables -t nat -A PREROUTING -d 192.168.56.102 -j DANT --to-destination
192.168.56.104

#PREROUTING实现限制来源和端口的转发
iptables -t nat -A PREROUTING -s 192.168.0.0/16 -p tcp --port 3333 -j DANT --to-
destination 192.168.56.104:22

#SNAT实现源地址转换
iptables -t nat -A POSTROUTING -s 172.16.0.0/16 -p tcp -j SNAT --to-source
192.168.56.104:22

#设置DNAT技术的完整实例（上面都是拆开的命令）
iptables -t nat -A PREROUTING -d 192.168.56.102 -p tcp --dport 3333 -j DANT --to-
destination 192.168.56.104:22
iptables -t nat -A POSTROUTING -d 192.168.56.104 -j SNAT --to-source
192.168.56.102
```

看最后的完整实例，数据包是有来有回的，你改了它的目标地址成了104后，数据包到达目标地址后要返回，这时数据包的来源还是106，但是它反不回去，从TCP连接建立上来说，106没给104发起过请求，发出的TCP请求还在和102建立着连接等待返回数据，所以返回数据必须是102到106，也就是数据包在104处理完后要回到102。故要设置两条iptables才能完成转发任务，这即是所谓的代理转发。

③FORWARD：

和先前提到的一样，它代表着Linux内核的路由和数据包转发，代表着iptables防火墙的FORWARD链。同一网段从一台机子到另一台机子的ping和它没关系，因为是直连路由，就算设计了FORWARD链也不影响互ping；分属不同网段的两台机子，互ping就有关了，因为需要防火墙充当路由器，它有路由功能；如果在防火墙本机上往外ping一个公网，不会用到它，这是直接把数据包从本机发出去，未用到路由功能。

只有需要路由功能和转发功能时，才和FORWARD有关。

④INPUT：

外来数据包进入机子，可设置运行/拒绝某IP的访问。没啥好说，待补充。

⑤OUTPUT

本机向外发送数据包，同上易理解没啥好说的，待补充。

6、iptablse拓展知识点补充

① -j 拒绝的动作有两种：DROP和REJECT。

如果是DROP拒绝，连接会卡住，并且没有任何返回提示，即是符合iptables防火墙规则的数据包直接丢弃，并且不告诉它返回信息。这时发起请求端一直不动（TCP/IP请求得不到返回），最后TIMEOUT才结束。

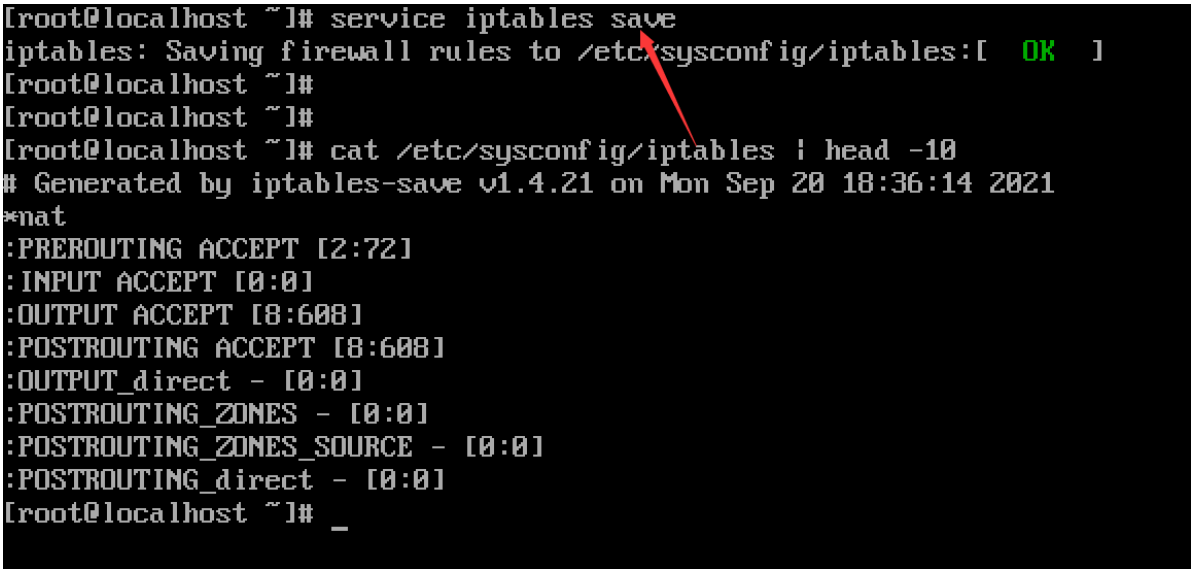
如果是REJECT拒绝，会得到一个回执，比较负责。

所以企业里对待恶意访问和攻击时，直接DROP；自己学习时，可以REJECT，便于调试规则。

② line-numbers 参数：如果要删除iptables里的某条规则，要先行号标注，再按行删除。

```
[root@localhost ~]# iptables -t nat -L -n --line-numbers | head -10
Chain PREROUTING (policy ACCEPT)
num  target     prot opt source               destination
1    PREROUTING_direct  all  --  0.0.0.0/0            0.0.0.0/0
2    PREROUTING_ZONES_SOURCE  all  --  0.0.0.0/0            0.0.0.0/0
3    PREROUTING_ZONES  all  --  0.0.0.0/0            0.0.0.0/0
4    DNAT       tcp  --  172.0.0.0/8          192.168.56.102       to:192.168.56.103

Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination

[root@localhost ~]# iptables -t nat -D PREROUTING 4
[root@localhost ~]# iptables -t nat -L -n --line-numbers | head -10
Chain PREROUTING (policy ACCEPT)
num  target     prot opt source               destination
1    PREROUTING_direct  all  --  0.0.0.0/0            0.0.0.0/0
2    PREROUTING_ZONES_SOURCE  all  --  0.0.0.0/0            0.0.0.0/0
3    PREROUTING_ZONES  all  --  0.0.0.0/0            0.0.0.0/0

Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
[root@localhost ~]# _
```

这是以nat表里的规则为例的删除操作，删除PREROUTING链里的第四条规则。



平时配置iptables，记得保存；在配置文件里删除或者添加某一行，更加规范。

7、iptables 4种标准跟踪状态

之前学习时，可以根据数据包源地址、目标地址、端口、协议等常用数据包描述信息来制定防火墙的规则。现在来看一种新的数据包描述信息，叫状态跟踪信息。

内核模块提供了一种数据包状态跟踪记录功能，指的是当数据包由防火墙本机发出去时，iptables会记录每个连接每个阶段的状态，并根据这个状态制定更细节的防火墙规则。

为什么要这玩意呢？在企业里，一台后端服务器上所设置的iptables规则往往时通过先设置INPUT链为默认拒绝状态，然后把个别的请求单独设置ACCEPT。然后，再加上OUTPUT链设置的开放，就可以起到比较好的防御作用。但是这样子，当这台机子向其他机子发起请求时，是行不通的，因为数据包有来有回，就算OUTPUT放出去了，回来时没设置到INPUT的ACCEPT，就不行了。那么多不同请求，我们岂不是要一一设置规则？太麻烦了！所以状态跟踪就发挥用途了，

①NEW状态：新建连接的第一个数据包。只要发出去一个数据包，就标记上NEW状态。

②ESTABLISHED状态：已经成功连接的数据包。数据包发送到对端，对端给出一个应答，应答标记上这个状态。

③RELATED状态：发出数据包相关的数据包。当前处于ESTABLISHED状态的连接再发起这个连接以外的又一个连接，就被标记成RELATED状态。一般来说，双通道（如FTP服务，建立一条通道后，在自己的基础上又开一条通道）会用到该状态，单通道（如ping、curl、Linux netstat -an TCP LINKS等，在一条通道上来回发送一次性往返）不用该状态。

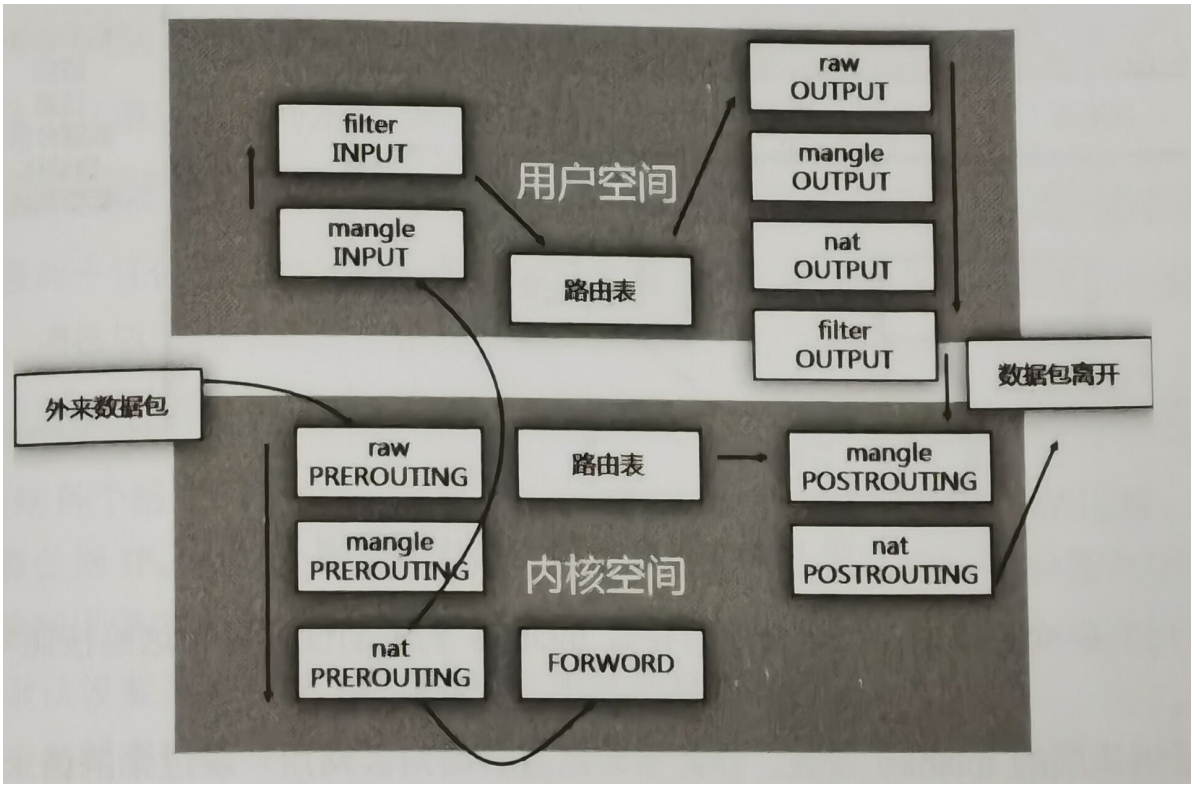
④INVALID状态：无效的数据包。不属于上面的状态的都是无效状态，该状态数据包通常直接被拒绝掉。

综上所述，在INPUT默认拒绝一起的情况下，无需个别设置INPUT，只需要输入 iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT，就可以让返回的数据包标记上ESTABLISHED状态，从而让机子接受它。如果用上RELATED，ping就会失败，因为ping这种ICMP是单通道，但FTP那些双通道就可以通过。

iptables -A INPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT ，一劳永逸。NEW可以不加，OUTPUT是默认允许的，故数据包出去时自带上NEW标记。INVALID的话，DROP即可，反正标记无效的，就别INPUT了。

8、iptables四表优先级和完整流程

优先级：raw——mangle——nat——filter，不管有没有设置、怎么设置规则，永远按这个优先级走，每到一个表，先看里面的规则，再去执行修饰语。



↑完整iptables流程图↑

9、iptables结合企业集群实际

针对服务器的网络攻击可谓道高一尺魔高一丈，现如今在服务器集群中，安全防护仰仗多方面的安全技术，iptables越来越趋向于一个辅助技术。

①最外层洪流层iptables设置。最外层的服务器直接面对公网用户发来的海量请求，因此要保证iptables默认对INPUT和FORWARD链式关闭的。

```
iptables -P INPUT DROR      #不要使用INPUT
iptables -P FORWARD DROP    #默认不允许任何形式的转发
iptables -P OUTPUT ACCEPT   #机子外送的包设置允许
```

至于端口的访问请求，是不允许公网用户访问类似SSH登录一类的重要端口，甚至是不能给发觉出该端口的存在，所以可以通过改变默认端口或者转发的形式，允许内部运维使用。

```
iptables -t filter -A INPUT -p tcp --dport xx -s "公司出口ISP地址" -j ACCEPT
```

对于特定的负载均衡窗口，要执行开放策略。

```
iptables -A INPUT -p tcp -dport 源端口:对外开放的端口 -j ACCEPT          #对外开发的端口
iptables -A FORWARD -d 10.172.100.0、24 -j ACCEPT                        #对负载代离后端的目标IP执行开放转发
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT    #开放本机发送请求的回应包
```

设置允许本机lo通信

```
iptables -t filter -A INPUT -i lo -j ACCEPT
iptables -t filter -A OUTPUT -o lo -j ACCEPT
```

允许内部的线下环境集群访问（如日志采集等）和禁止公网外来的ping

```
iptables -A INPUT -s 10.171.100.0/24 -j ACCEPT
iptables -A INPUT -p icmp -j DROR
```

②对于后端各个层面的iptables设置，由于一般不会设置公网IP，所以轻松很多，默认规则是：INPUT默认设置为REJECT，且只允许上层的内网地址通过；INPUT开启状态追踪以接收更后端的返回包；允许线下集群的访问。

```
iptables -P INPUT REJECT    #reject防止上层内网的请求卡住
iptables -A INPUT -s 10.173.100.0/24 -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT  #开放本机发送请求的回应包
iptables -A INPUT -s 10.171.100.0/24 -j ACCEPT  #允许线下集群连接
FORWARD, OUTPUT
```

10、注意的地方

①配置iptables尤其是INPUT链时，清空已有规则时要先 -p INPUT ACCEPT ，否则因为先前设置DROP为默认，-f 后允许的规则被删了，导致自己被挡出去了。如果要清空，尽量用 service iptables stop ，这样默认规则会被指定为ACCEPT，但是注意其他表的临时规则也会被清空，要好好保存本地。

②配置防火墙规则时，要注意越是细致严格的规则越靠前，越是笼统的越靠后

- 后记：

真寄吧难，这一节学麻了，反反复复看了两遍只能懂个70%，模糊的地方也没办法了。