iptables

netfilter/iptables（简称iptables）是与2.4.x和2.6.x系列版本Linux内核集成的IP信息包过滤系统。

Iptables Tutorial

1、表和链

1.1、表

iptables会根据不同的数据包处理功能使用不同的规则表。它包括如下五个表：`filter`、`nat`和`mangle`,`raw`,`security`。

- `filter`

  `filter`是默认的表(如果命令中没有使用`-t`指定表，就会使用`filter`)，包含真正的防火墙过滤规则。

内建的规则链包括：`INPUT`（处理进入的数据包）、`OUTPUT`（处理本地生成的数据包）和`FORWARD`（处理转发的数据包）。

在filter表中只允许对数据包进行DROP或ACCEPT操作，而无法对数据包进行更改。

- `nat`

  `nat`表主要用于进行网络地址转换（Network Address Translation, NAT）。包含源地址、目的地址及端口转换使用的规则，当遇到创建新连接的数据包时，会查阅此表。

内建的规则链包括`PERROUTING`、`OUTPUT`和`POSTROUTING`。

`PREROUTING`链的作用是在包刚刚到达防火墙时改变它的目的地址（如果需要的话）。

`OUTPUT`链改变本地产生的包的目的地址。

`POSTROUTING`链在包就要离开防火墙之前改变其源地址，此表仅用于`NAT`，也就是转换包的源或目标地址。实际的操作分为以下几类：

（1）DNAT:主要用在这样一种情况，即假设你有一个合法的IP地址，要把对防火墙的访问重定向到其他的机器上（比如DMZ）。也就是说，我们改变的是目的地址，以使包能重路由到某台主机。

（2）SNAT:SNAT改变包的源地址，这在极大程度上可以隐藏本地网络或者DMZ等。一个很好的例子是我们知道防火墙的外部地址，但必须用这个地址替换本地网络地址。有了这个操作，防火墙就能自动地对包做SNAT和De-SNAT（就是反向的SNAT），以使LAN能连接到Internet。

- `mangle`

  `mangle`表主要用来定义数据包的操作方式。我们可以改变不同的包及包头的内容，比如TTL、TOS或MARK。这些标志随后被filter表中的规则检查。

  内建的规则链包括：`PREROUTING`、`INPUT`、`FORWARD`、`POSTROUTING`和`OUTPUT`。

❑ `raw`表设置raw一般是为了不再让iptables做数据包的连接跟踪处理提高新能。内建的规则链包括：`PREROUTING`和`OUTPUT`。

❑ `security`表用于强制访问控制 （MAC） 网络规则，例如由 `SECMARK` 和 `CONNSECMARK` 目标启用的规则。 强制访问控制由 Linux Security Mod- 实现诸如SELinux之类的Ules。

1.2、链

链是数据包传播的路径，一条链就是规则的一个检查清单，每一条链中可以有一条或多条规则。当一个数据包到达一个链时，iptables就会从链中第一条规则开始检查，查看该数据包是否满足规则所定义的条件，决定是否按预定义的方法处理该数据包，如果包头不符合链中的规则，iptables就会根据该链的默认策略来处理数据包。

表对应的相关规则链的功能如下:

❑ INPUT链：当一个数据包由内核中的路由计算确定为本地的Linux系统后，它会通过INPUT链的检查。

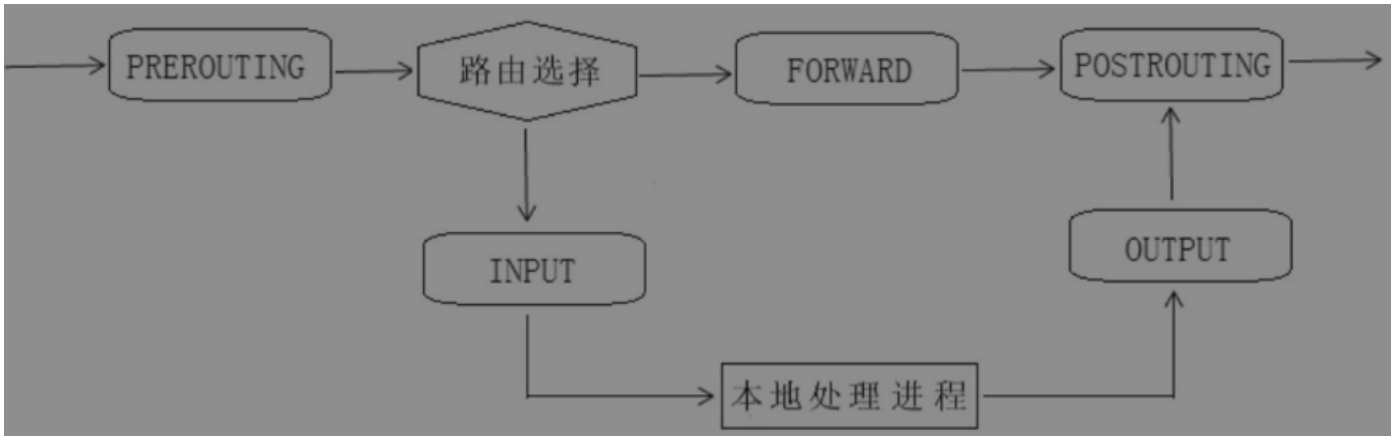
❑ OUTPUT链：保留给系统自身生成的数据包。

❑ FORWARD链：经过Linux系统路由的数据包（即当iptables防火墙用于连接两个网络时，两个网络之间的数据包必须流经该防火墙）。

❑ PREROUTING链：用于修改目的地址（DNAT）。

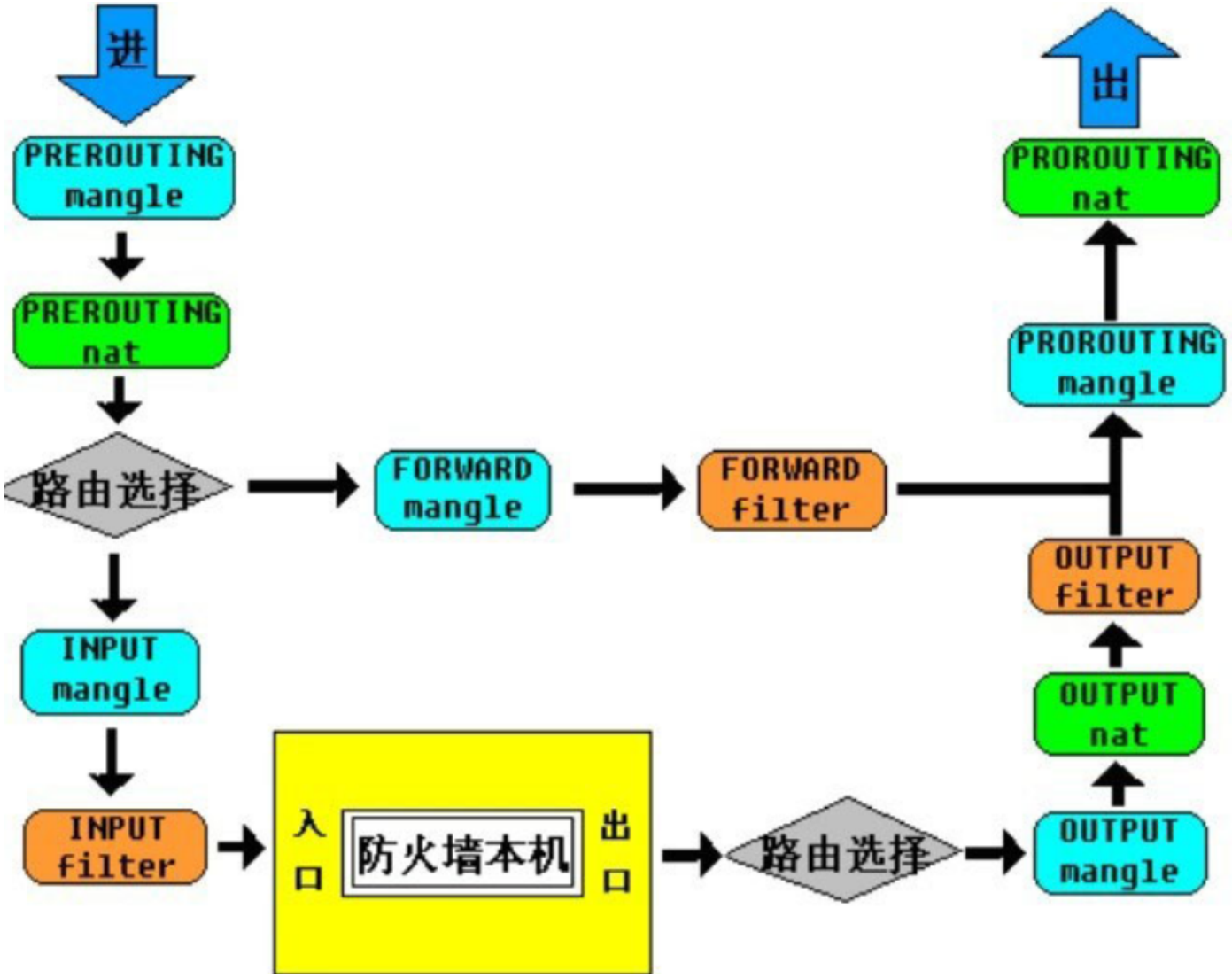
❑ POSTROUTING链：用于修改源地址（SNAT）。

1.3、iptables五条链的相互关系。



1.4 iptables传输数据包的过程

数据包流经`iptables`防火墙的路径是经过严格定义的一个处理过程，下图描述了数据包流经`iptables`防火墙时的传输过程。



（1）流入本机的数据包穿过iptables防火墙的传输过程

官方文档位置

过程1：数据包从网络传入，并由网卡接收。

过程2：随后转入mangle表的PREROUTING链。

过程3：再转入nat表的PRETOUTING链，这个链主要用来做DNAT，即目的地址转发。

过程4：内核对数据包进行路由选择。

过程5：因为数据包是传入本机的，因此转入mangle表的INPUT链。

过程6：然后转入filter表的INPUT链。

过程7：最终到达接收数据包的应用程序。

（2）流出本机的数据包穿过iptables防火墙的传输过程

官方文档位置

过程1：应用程序生成数据包，根据源地址、目的地址、外出接口等信息进行路由判断。

过程2：随后转入mangle表的OUTPUT链。

过程3：再转入nat表的OUTPUT链，这个链可以用来做DNAT。

过程4：进入filter表的OUTPUT链，对该数据包进行选择性过滤。

过程5：然后进入mangle表的POSTROUTING链。

过程6：进入nat表的POSTROUTING链，该链可以做SNAT，即源地址转发，最终进入网络。

（3）流经本机转发的数据包的传输过程

官方文档位置

过程1：数据包从网络传入，并由网卡接收。

过程2：随后转入mangle表的PREROUTING链。

过程3：再转入nat表的PRETOUTING链，这个链主要用来做DNAT，即目的地址转发。

过程4：内核对数据包进行路由选择。该包的目的地址是另一台主机，所以转入mangle表的FORWARD链。

过程5：再转入filter表的FORWARD链，针对这类包的所有过滤操作都在该链进行。

过程6：过滤后转入mangle表的POSTROUTING链。

过程7：最后通过nat表的POSTROUTING链进行SNAT，最终进入网络。

> （1）用户可以在各个链定义规则。当数据包到达上图的任意一个链时，iptables就会根据链中定义的规则来处理这个数据包。iptables将数据包的头信息与它所传递到链中的每条规则进行比较，看它是否与某条规则完全匹配。如果数据包与某条规则匹配，iptables就对该数据包执行由该规则指定的操作。如果某条链中的规则决定要丢弃（DROP）数据包，数据包就会在该链中丢弃；如果链中规则接收（ACCEPT）数据包，数据包就可以继续前进。但是，如果数据包和某条规则不匹配，那么它将与链中的下一条规则进行比较。如果该数据包不符合该链中的任意一条规则，那么iptables将根据该链预先定义的默认策略来决定如何处理该数据包。
>
> （2）`PREROUTING`和`POSTROUTING`链只对请求连接的数据包进行操作，对属于该连接的后续数据包，不予比对规则，只按已确定的规则自动进行操作。因此建议不要在此链上作过滤操作；否则将漏掉对后续数据包的过滤。

2、iptables命令格式

iptables命令的基本格式如下：<>括起来的为必设项，[]括起来的为可选项。iptables命令要求严格区分大小写。

```
iptables [-t table] <COMMAND>  [chains] [rule-matcher] [ -j target ]
```

各选项说明：

- 表选项（table）

  `netfilter`的表操作是以-t或--table

  来指定的，未指定时默认为filter表。table选项的参数.

参　数	描　述
filter	过滤表
nat	网络地址转换表
mangle	数据包处理表

- 常用操作命令选项（COMMAND）

- 链选项（chains）

> ➢filter表有INPUT、OUTPUT、FORWARD和自定义4种链形式。
>
> ➢nat表有OUTPUT、PREROUTING和POSTROUTING三种链形式。
>
> ➢mangle表有INPUT、OUTPUT、FORWARD、PREROUTING和POSTROUTING五种链形式。

- 常用匹配规则选项（rule-matcher）

  匹配规则选项指定数据包与规则匹配所应具备的特征，包括源地址、目的地址、传输协议（如TCP、UDP、ICMP等）和端口号等。

> “! ”为逻辑非；接口名后跟“+”表示所有以此接口名开头的接口都会被匹配。下图为匹配条件扩展。

- 目标动作选项（target）

  当规则匹配一个包时，要执行的目标动作以-j参数标识

  - filter表的目标动作

  - nat表的目标动作

> SNAT用于进行源网络地址转换，这意味着该目标将重写包的IP头中的源IP地址。这就是我们想要的，
>
> 例如，当几个主机共享一个互联网连接时。然后，我们可以在内核中打开ip转发，并编写一个SNAT规则，将从本地网络发出的所有数据包转换为我们自己的互联网连接的源ip。如果不这样做，外界将不知道向哪里发送应答包，因为我们的本地网络大多使用分配给局域网网络的IANA指定的IP地址。如果我们按原样转发这些数据包，互联网上就没有人知道它们实际上是来自我们的。SNAT目标完成完成这类工作所需的所有翻译工作，让所有离开我们局域网的数据包看起来就像来自单一主机，即我们的防火墙。
>
> SNAT 只在nat表内有效，在POSTROUTING 链内有效。
>
> SNAT参数`--to-source`
>
> 例如：`iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to-source 194.236.50.155:1024`
>
> `--to-source` 选项用于指定数据包应使用的源。修改数据包的源IP和端口号。如果是个范围，也可以这样写:`194.236.50.155-194.236.50.160:1024-32000`

DNAT 用于执行目标网络地址转换，这意味着它用于重写数据包的目标 IP 地址。如果数据包匹配，并且这是规则的目标，则数据包和同一流中的所有后续数据包将被转换，然后路由到正确的设备、主机或网络。

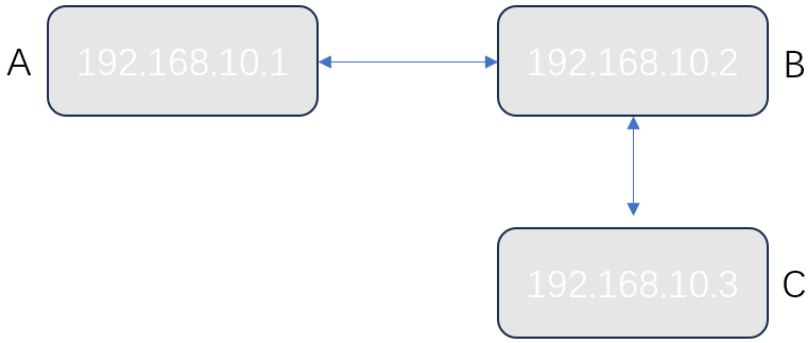
DNAT仅在nat表中的`PREROUTING`和`OUTPUT`链

DNAT参数`--to-destination`

例如：`iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT --to-destination 192.168.1.1-192.168.1.10`

`--to-destination`选项--to-destination 选项告诉 DNAT 机制要在 IP 标头中设置哪个目标 IP，以及将匹配的数据包发送到何处。上面的例子将把所有以15.45.23.67为目的IP地址、目的端口号80的分组发送到局域网IP范围内，即192.168.1.1到10。例如，规则可以是——to-destination 192.168.1.1:80，如果我们想指定一个端口范围，规则可以是——`to-destination 192.168.1.1:80-100`。

注意： SNAT和DNAT一般是要成对使用的。



如上图，B做了DNAT的转发，会把消息转发给C服务器。

A、B、C三个服务器。A发请求到B，B修改目的IP为192.168.10.3，将请求转发到了C。C处理完后，回复消息，这时C发出的消息的源IP是它自己的IP 192.168.10.3。A服务器收到消息后，发现源IP不是它请求的B的IP地址，会将数据直接丢弃。

所以需要同时修改C发出消息的源IP为B的IP地址。

REDIRECT目标用于将分组和流重定向到计算机本身。这意味着，例如，我们可以将所有发送到HTTP端口的数据包重定向到我们自己主机上的HTTP代理，如squid。本地生成的数据包映射到127.0.0.1地址。换句话说，这将为转发的数据包或类似内容重写到我们自己的主机的目标地址。重定向目标非常适合在我们需要的时候使用，例如，透明代理，局域网主机根本不知道代理。

DNAT参数`--to-destination`

例如：`iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080`

--to-ports 选项指定要使用的目标端口或端口范围。如果没有 --to-ports 选项，则永远不会更改目标端口。上面的命令就是将tcp端口号为80的数据包重定向到8080端口。

- mangle表的目标动作

- 扩展的目标动作

> 要使用扩展的目标动作，必须在内核中激活相应选项或装载相应内核模块。

3、 iptables的状态state

iptables防火墙的状态（state）：

❑ NEW：如果你的主机向远程机器发出一个连接请求，这个数据包的状态是NEW。

❑ ESTABLISHED：在连接建立之后（完成TCP的三次握手后），远程主机和你的主机通信数据的状态为ESTABLISHED。

❑ RELATED：和现有联机相关的新联机封包。像FTP这样的服务，用21端口传送命令，而用20端口（port模式）或其他端口（PASV模式）传送数据。在已有的21端口上建立好连接后发送命令，用20或其他端口传送的数据（FTP-DATA），其状态是RELATED。

❑ INVALID：无效的数据包，不能被识别属于哪个连接或没有任何状态，通常这种状态的数据包会被丢弃。

如规则`iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT`表示接受`ESTABLISHED,RELATED`状态的数据包。

如规则`iptables -A INPUT -m state --state NEW -j DROP`这个规则是将所有发送到你机器上的数据包（状态是`NEW`的包）丢弃，也就是不允许其他的机器主动发起对你的机器的连接，但是你却可以主动连接其他的机器。

4、 iptables的使用

- 规则`iptables -P INPUT DROP`会将进入主机的所有数据全部丢掉。如果你是通过远程shell执行这个命令，那么执行后shell将会断开。

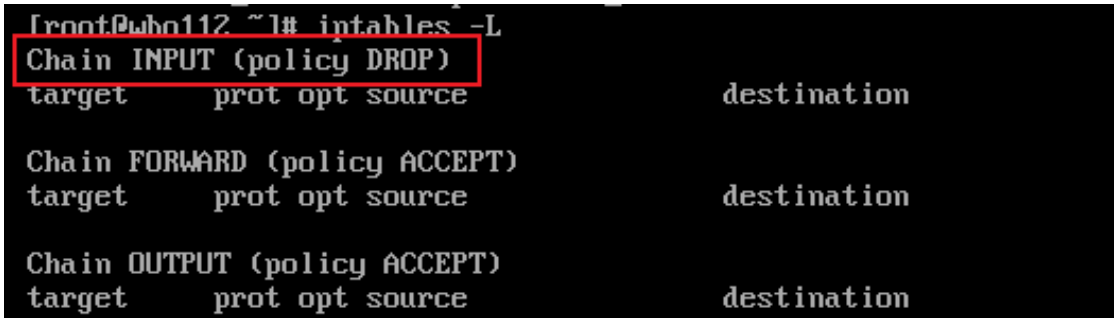
比如我的主机当前的规则，所有的都是策略都是`ACCEPT`

```
[root@wbo112 ~]# iptables -L
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
```

当执行`iptables -P INPUT DROP`后，远程`shell`就会断开，这时通过主机内的shell再次查看当前的规则，就能看到`INPUT`变成了`DROP`。

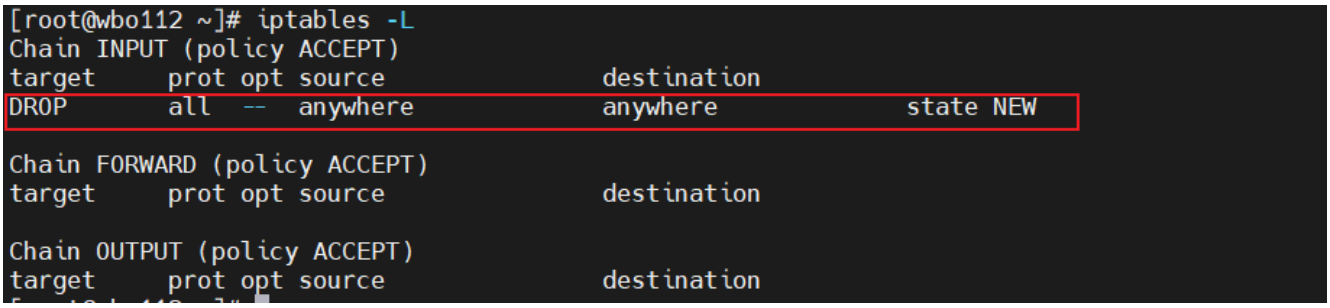


再次执行`iptables -P INPUT ACCEPT`将`ipables`恢复。

- 规则 `iptables -A INPUT -m state --state NEW -j DROP`

  这个规则是将所有发送到你机器上的数据包（状态是`NEW`的包）丢弃，也就是不允许其他的机器主动发起对你的机器的连接，但是你却可以主动连接其他的机器，不过仅仅是连接而已，连接之后的数据是`ESTABLISHED`状态的。

  执行上面的规则后，在查看当前的规则，就能看到下面`DROP`的一条。



这时，如果你本地再新建一条远程shell去连接主机，就会发现无法连接。而之前已经连接的远程shell是可以正常使用的。

2. 如果安装了nf_conntrack模块，可以查看nf_conntrack记录，命令如下：

```
1         cat /proc/net/nf_conntrack
```

老版的iptables的conntrack称为ip_conntrack，新版的名为nf_conntrack。nf_conntrack支持IPv4和IPv6，而ip_conntrack只支持IPv4。

不建议在线上服务器中开启iptable的Conntrack功能。

Conntrack模块维护的所有信息都包含在这个例子中了，通过它们就可以知道某个特定的连接处于什么状态。首先显示的是协议，这里是tcp，接着是十进制的（tcp的协议类型代码是6）。之后的117是这条conntrack记录的生存时间，它会有规律地被消耗，直到收到这个连接更多的包。到那个时候，该值就会被设为当时那个状态的默认值。接下来的是这个连接在当前时间点的状态。上面的例子说明了这个包处于状态SYN_SENT下，这个值是iptables显示的，便于我们理解，而内部用的值稍有不同。SYN_SENT说明我们正在观察的这个连接只在一个方向发送了一个TCP SYN包。再下面是源地址、目的地址、源端口和目的端口。其中有个特殊的词UNREPLIED，说明这个连接还没有收到任何回应。最后，是希望接收的应答包的信息，它们的地址和端口与前面相反。

5、保存iptables规则

`iptables-save > /opt/iptables_save` 将当前的iptables配置保存到 /opt/iptables_save中

`iptables-restore < /opt/iptables_save` 从/opt/iptables_save中恢复iptables的配置

6、一些常用的命令

> 注意这里没有使用`-t`参数指定表名，默认显示的就是`filter`表，如果要显示其他的表，需要使用`-t`后加表名。如`-t nat`。

- 查看iptables规则

```
iptables -L
```

> -L参数可以后跟`--line-numbers`参数打印出行号。
>
> 如`iptables -L --line-numbers`。后面命令中指定插入行号，替换行号等等都可以参照`--line-numbers`输出的行号

- 将IP地址和端口号以数字格式显示列出所有链的规则。

```
iptables -nL
```

- 详细列出所有链的规则

```
iptables -vL
#不能使用 iptables -Lv
```

- 列出INPUT链的规则

```
iptables -L INPUT
```

- 列出INPUT链的1号规则

```
iptables -L INPUT 1
```

- 显示所有链的规则

```
iptables -S
```

- 详细显示所有链的规则

```
iptables -vS
#不能使用iptables -Sv
```

- 显示INPUT链的规则

```
iptables -S INPUT
```

- 显示INPUT链的1号规则

```
iptables -S INPUT 1
```

清除指定链和表中的所有规则

- 清除所有链的规则（默认为filter表）

```
iptables -F
```

- 清楚INPUT链的所有规则

```
iptables -F INPUT
```

- 将所有链中的规则的包字节计数器清零。

```
iptables -Z
```

- 将INPUT链中的规则的包字节计数器清零

```
iptables -Z INPUT
```

- 在INPUT、OUTPUT和FORWARD链上设置默认规则策略为DROP（拒绝所有数据包）

```
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
```

- 在INPUT链上添加规则，协议为tcp，目标端口号是21

```
iptables -A INPUT -p tcp --dport 21
```

这里没有指定动作，所以就按照`INPUT`的默认动作进行处理。

执行完上面的命令再次查看，就会看到`INPUT`中多了一个对应的条目

```
[root@wbo112 ~]# iptables -A INPUT -p tcp --dport 21
[root@wbo112 ~]# iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    tcp  --  anywhere             anywhere             tcp dpt:ftp

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

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
```

- 在INPUT链上插入规则，协议为tcp，目标端口号是22

```
iptables -I INPUT 1 -p tcp --dport 23
```

从下图也能看到多了一条dpt:telnet的条目，原来的dpt:ftp的行号已经变成了2

```
[root@wbo112 ~]# iptables -I INPUT 1 -p tcp --dport 23
[root@wbo112 ~]# iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    tcp  --  anywhere             anywhere             tcp dpt:telnet
2    tcp  --  anywhere             anywhere             tcp dpt:ftp

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

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
```

- 在INPUT链上替换规则号1的iptables规则，将目标端口号更改为24

```
iptables -R INPUT 1 -p tcp --dport 24
```

```
[root@wbo112 ~]# iptables -R INPUT 1 -p tcp --dport 24
[root@wbo112 ~]# iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    tcp  --  anywhere             anywhere             tcp dpt:lmtp
2    tcp  --  anywhere             anywhere             tcp dpt:ftp

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

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
```

- 删除规则

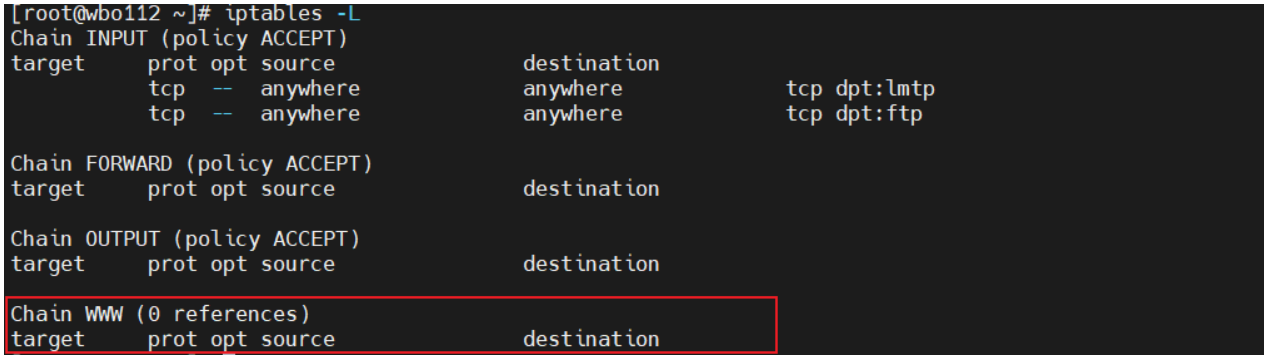
首先找到规则对应的行号。使用`-L --line-numbers`参数。

比如要删除上面INPUT中的tcp端口是24的规则。通过`iptables -L --line-numbers`看到对应行号是1。

执行`iptables -D INPUT 1`就可以删除。

- 创建用户自定义链

```
iptables -N WWW   #创建用户自定义链WWW
```



- 指定协议

```
iptables -A INPUT -p tcp -j ACCEPT
iptables -A INPUT -p udp -j ACCEPT
```

- 指定ICMP类型

```
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
```

- 指定IP地址

```
iptables -A INPUT -s 192.168.0.5   -j ACCEPT
iptables -A INPUT -s 192.168.1.0/24   -j ACCEPT  #允许192.168.1.0这个子网的所有主机访问
```

- 指定接口

```
iptables -A INPUT -i eth0 -j ACCEPT
iptables -A FORWARD -o eth0 -j ACCEPT
iptables -A FORWARD -o ppp+ -j ACCEPT
```

- 指定端口号

```
iptables -A INPUT -p tcp --sport www -j ACCEPT
iptables -A INPUT -p tcp --sport 80-j ACCEPT
iptables -A INPUT -p tcp --dport 53-j ACCEPT
iptables -A INPUT -p tcp --sport 22:80-j ACCEPT
```

##

如果要实现NAT功能，首先需要将文件“/proc/sys/net/ipv4/ip_forward”设置为1（默认是0），才能打开内核的路由功能。

具体命令如下：

```
echo "1">/proc/sys/net/ipv4/ip_forward
```