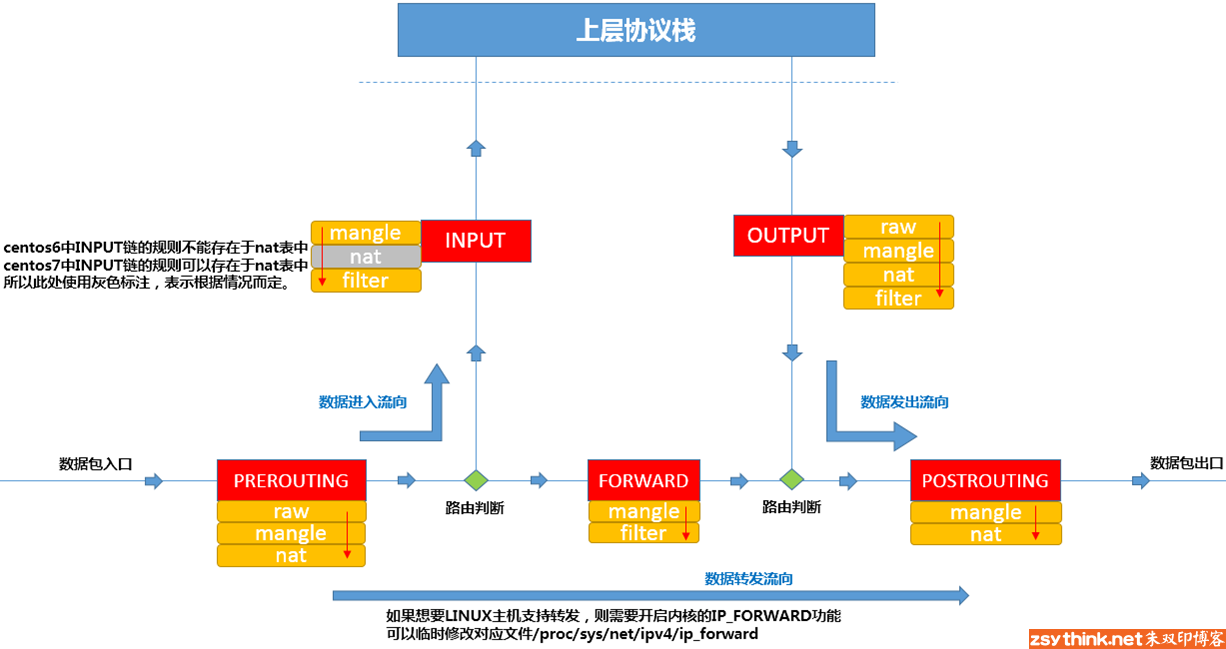
# IPTABLES基础命令

韦体东

本文主要参考 <http://www.iptables.info> 和 <http://www.zsythink.net/archives/tag/iptables/>，不是详细的原理介绍，权当个人记录的操作笔记。

## 基本概念

借个图看一下，这个图很熟悉了，正是netfilter框架的hook point：



Iptables是基于netfilter做的，有几个概念：

1. 表一共有filter, nat, mangle, raw四个，用-t参数指定：

filter表，用于做包过滤，可以在此表中执行DROP,LOG,ACCEPT,REJECT动作。如果没有指定-t，即默认为此表。

nat表，主要用于做NAT规则转换，一条流上的包，只有首包才会\*匹配\*此表中的NAT规则，后继包不需要匹配此表，会跟着自动转换。

mangle表，主要用于修改包内容或者包头，比如修改TTL，TOS，MSS等，或者给包大一个MASK(并非打在包数据上，而是mbuf上)以供后续规则使用。

raw表，是2.6以后新加的，他负责在其他表生效前生效，raw的意思其实就是没有经过任何iptables规则染指过的输入数据和输出数据。最开始输入，其位于PREROUTING前，没有被污染的输出，当然是OUTPUT前。

2. chain

链，如上图，PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING就是默认链，也就是规则生效的HOOK POINT，而用户也可以自定义链，自定义并不是超越这几个链创建了新的HOOK POINT，而是做了一个规则的汇聚。

3. targets和jumps：

-j参数指定的是targets或者jumps，默认内置了很多targets，例如：

ACCEPT, CLASSIFY, CLUSTERIP, CONNMARK, CONNSECMARK, DNAT, DROP, DSCP, ECN, LOG options, MARK, MASQUERADE, MIRROR, NETMAP, NFQUEUE, NOTRACK, QUEUE, REDIRECT, REJECT, RETURN, SAME, SECMARK, SNAT, TCPMSS, TOS, TTL, ULOG

jumps是指自定义链的名称，创建自定义链的目的是为了方便管理，就像编程里面有了语句，也希望有函数，通过一个函数名，可以引用函数内的多条语句，通过-j转到自定义链。

## 重置规则

基本来说，做实验前，需要排除之前设置的iptables干扰，使用-F可以清空iptables规则，如果不指定-t，都表示在filter表中(-t filter)

do ~ # iptables -F

-F清空的是默认规则，如果想清空自定义链规则，需要用-X，后面再介绍。这里有个重置iptables规则的命令集，执行即可：

iptables -t nat -F

iptables -t nat -X

iptables -t nat -P PREROUTING ACCEPT

iptables -t nat -P POSTROUTING ACCEPT

iptables -t nat -P OUTPUT ACCEPT

iptables -t mangle -F

iptables -t mangle -X

iptables -t mangle -P PREROUTING ACCEPT

iptables -t mangle -P INPUT ACCEPT

iptables -t mangle -P FORWARD ACCEPT

iptables -t mangle -P OUTPUT ACCEPT

iptables -t mangle -P POSTROUTING ACCEPT

iptables -F

iptables -X

iptables -P FORWARD ACCEPT

iptables -P INPUT ACCEPT

iptables -P OUTPUT ACCEPT

iptables -t raw -F

iptables -t raw -X

iptables -t raw -P PREROUTING ACCEPT

iptables -t raw -P OUTPUT ACCEPT

## 查看规则

清空后，默认规则是ACCEPT，所以任何链接都不会被阻拦或丢弃。使用-L查看某表的规则，例如查看filter表的规则(其他表请自行加-t参数，后面不再赘述)：

do ~ # iptables --line-number -L -nv

Chain INPUT (policy ACCEPT 110 packets, 23541 bytes)

num pkts bytes target prot opt in out source destination

n和linux大多数网络命令的n类似，表示不解析地址和端口；而v代表verbose输出；--line-number加上后，规则会显示num一列，方便删除的时候使用num序号删除。

查看所有表规则：

iptables --line-number -L -nv

iptables --line-number -L -nv -t nat

iptables --line-number -L -nv -t raw

iptables --line-number -L -nv -t mangle

## 增删改规则

-A表示APPEND，后面的参数是“链”名称，-A是在一个链的后面添加；如果用-I，表示INSERT，是在一个链的前面插入；如果用-R，表示REPLACE，是替换指定num序号的规则；-D表示DELETE指定规则。

例如，APPEND一条DROP来自源IP为 200.200.99.190 的包：

do ~ # iptables -t filter -A INPUT -s 200.200.99.190 -j DROP

do ~ # iptables --line-number -L -nv

Chain INPUT (policy ACCEPT 183 packets, 55557 bytes)

num pkts bytes target prot opt in out source destination

1 2 168 DROP all -- \* \* 200.200.99.190 0.0.0.0/0

现在，你从源IP 200.200.99.190 ping一下do这台机，窗口是没有不回显消息的。

现在，我们尝试将刚才的DROP规则替换成REJECT，因为num序号显示其为第一条规则，于是-R INPUT后指定1即可：

do ~ # iptables -t filter -R INPUT 1 -s 200.200.99.190 -j REJECT

do ~ # iptables --line-number -L -nv

Chain INPUT (policy ACCEPT 95 packets, 23037 bytes)

num pkts bytes target prot opt in out source destination

1 2 168 REJECT all -- \* \* 200.200.99.190 0.0.0.0/0 reject-with icmp-port-unreachable

记得，此处的REJECT其实是可以配合--reject-with选项，其值有：

icmp-net/host/port/proto-unreachable

icmp-net/host/admin-prohibited

如果不使用此选项，默认是icmp-port-unreachable，所以，你在此从源IP 200.200.99.190 ping do这台机时候，会显示：

From 200.200.96.29 icmp\_seq=332 Destination Port Unreachable

上面修改DROP为REJECT后，我们来参数-D删除第一条规则，在-D 表明后面需要跟上规则序号，例如第1条规则：

do ~ # iptables -D INPUT 1

do ~ # iptables --line-number -L -nv

Chain INPUT (policy ACCEPT 254 packets, 84782 bytes)

num pkts bytes target prot opt in out source destination

## 默认规则修改

FORWARD的默认规则为ACCEPT

do ~ # iptables --line-number -L -nv

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)

num pkts bytes target prot opt in out source destination

如果我们想修改为DROP，可以使用-P参数：

do ~ # iptables -t filter -P FORWARD DROP

do ~ # iptables --line-number -L -nv

Chain FORWARD (policy DROP 0 packets, 0 bytes)

num pkts bytes target prot opt in out source destination

## 源地址和目标地址

-s指定源地址，-d指定目标地址，两者都可以用逗号分隔多个源或者目标，也可以对该条件取反，用 ! -s , ! –d表示(取反后不能用逗号操作)，例如 -s 192.168.10.100,192.168.10.122表示来自192.168.10.100或者192.168.10.122的数据，! -s 192.168.10.100表示，除了来自192.168.10.100的数据。假设，我们想要禁止所有的ping包，但是来自200.200.99.190的例外(TCP之类的协议不会受影响)，可以：

do ~ # iptables -A INPUT ! -s 200.200.99.190 -p icmp -j REJECT

do ~ # iptables --line-number -L -nv

Chain INPUT (policy ACCEPT 974 packets, 267K bytes)

num pkts bytes target prot opt in out source destination

1 11 924 REJECT icmp -- \* \* !200.200.99.190 0.0.0.0/0 reject-with icmp-port-unreachable

## 输入和输出网口

-i和-o用于指定入网口和出网口，鉴于有的链上只有输入没有输出，可以理解-i只能适用(PREROUTING INPUT FORARD)链，-o只能用于(FORWARD OUTPUT POSTROUTING)链。

例如，想要禁止来自eth3口的所有ICMP可以：

do ~ # iptables -A INPUT -i eth3 -p icmp -j REJECT

同源地址目标地址类似，输入输出网口也可以取反。

## 扩展模块tcp

适用-m参数指定扩展模块名称，有的功能是由非内置提供的，如匹配tcp链接的某端口，就需要-m tcp这个扩展模块，例如，拒绝来自 200.200.90.100 的ssh链接(ssh连接端口22)：

do ~ # iptables -A INPUT -s 200.200.90.100 -p tcp -m tcp --dport 22 -j REJECT

do ~ # iptables --line-number -L -nv

Chain INPUT (policy ACCEPT 158 packets, 38691 bytes)

num pkts bytes target prot opt in out source destination

1 0 0 REJECT tcp -- \* \* 200.200.90.100 0.0.0.0/0 tcp dpt:22 reject-with icmp-port-unreachable

在200.200.90.100上再次链接会显示：

# ssh root@200.200.96.29 ls

ssh: connect to host 200.200.96.29 port 22: Connection refused

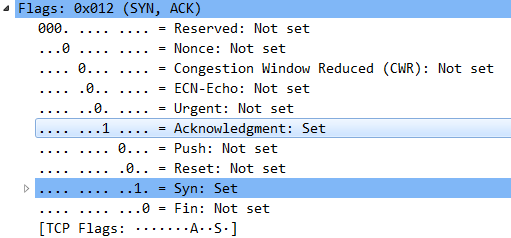
对于已有链接，同样会断开：

# ssh root@200.200.96.29

do ~ #

do ~ # Write failed: Broken pipe

如果我们只想禁止新链接，对于原有链接放通，我们可以使用tcp扩展模块的--tcp-flags参数。Tcp-flags定义的是TCP包的头部12个bit的flags：



--tcp-flags使用SYN,ACK,FIN,RST,URG,PSH等来表示这些位。

--tcp-flags需要指定两个参数，第一个参数，是所有要匹配的位，第二个参数是在这些匹配位中，哪些位需为1，如果没在第二个参数里面出现，表示需为0.

例如，拒绝新建链接(老的链接依然可用)，可以匹配SYN=1的包：

do ~ # iptables -A INPUT -s 200.200.90.100 -p tcp -m tcp --dport 22 --tcp-flags SYN,ACK SYN -j REJECT

do ~ # iptables --line-number -L -nv

Chain INPUT (policy ACCEPT 419 packets, 135K bytes)

num pkts bytes target prot opt in out source destination

1 0 0 REJECT tcp -- \* \* 200.200.90.100 0.0.0.0/0 tcp dpt:22 flags:0x13/0x02 reject-with icmp-port-unreachable

当然，有个更简单的写法 -m tcp --sync效果一样。

第一个参数如果是想匹配所有flags可以用ALL替代。

当使用-p指定协议的时候，如果没有指定-m，会自动加载和-p同名的模块来作为扩展模块，所以，如果要指定--dport 22，省略掉-m tcp也可以。

--dport和--sport都可以用端口范围表示，例如：

--dport 22:25

当然，有tcp模块也有udp模块，都有sport和dport。

## 扩展模块multiport

如果想指定多个离散的端口，需要multiport模块：

-p tcp -m multiport --dports 22,139,445

如果-d -s想匹配一个连续的IP段，需要iprange模块:

--src-range --dst-range

## 扩展模块string

string模块，可以指定 --algo bm|kmp 来选择匹配算法，--string 来指定匹配的字符串

time模块，可以用--timestart来指定起始时间--timestop来指定停止时间， --weekdays 6,7来指定周末，比如周末不能上网，--monthdays 22,23来指定一个月中的日期

## 扩展模块connlimit

connlimit模块，可以限定IP地址同时连接server的链接数，例如 --connlimit-above 2 -j REJECT，有两个并发就拒绝，--connlimit-mask 24 表示匹配某类网段的链接数量，24表示C类网络地址。

limit模块，对报文速率做限制。例如，5s一个ping包可以这样配置：

do ~ # iptables -A INPUT -s 200.200.90.100 -p icmp -m limit --limit-burst 1 --limit 12/minute -j ACCEPT

do ~ # iptables -A INPUT -s 200.200.90.100 -p icmp -j REJECT

do ~ # iptables --line-number -L -nv

Chain INPUT (policy ACCEPT 460 packets, 98917 bytes)

num pkts bytes target prot opt in out source destination

1 0 0 ACCEPT icmp -- \* \* 200.200.90.100 0.0.0.0/0 limit: avg 12/min burst 1

2 0 0 REJECT icmp -- \* \* 200.200.90.100 0.0.0.0/0 reject-with icmp-port-unreachable

序号从小往大匹配，如果匹配到12/minute（5的倍数个）直接放通，如果没有匹配上，会匹配第二条规则，第二条规则是所有ICMP REJECT

limit-burst先给限制1个令牌，保证第一个包能通，如果你想保证前3个包能通，需要给3个令牌，默认是5个令牌，不能设置为0。

# ping 200.200.96.29

PING 200.200.96.29 (200.200.96.29) 56(84) bytes of data.

64 bytes from 200.200.96.29: icmp\_req=1 ttl=61 time=1.05 ms

From 200.200.96.29 icmp\_seq=2 Destination Port Unreachable

From 200.200.96.29 icmp\_seq=3 Destination Port Unreachable

From 200.200.96.29 icmp\_seq=4 Destination Port Unreachable

From 200.200.96.29 icmp\_seq=5 Destination Port Unreachable

64 bytes from 200.200.96.29: icmp\_req=6 ttl=61 time=0.997 ms

From 200.200.96.29 icmp\_seq=7 Destination Port Unreachable

From 200.200.96.29 icmp\_seq=8 Destination Port Unreachable

From 200.200.96.29 icmp\_seq=9 Destination Port Unreachable

From 200.200.96.29 icmp\_seq=10 Destination Port Unreachable

64 bytes from 200.200.96.29: icmp\_req=11 ttl=61 time=0.970 ms

From 200.200.96.29 icmp\_seq=12 Destination Port Unreachable

From 200.200.96.29 icmp\_seq=13 Destination Port Unreachable

From 200.200.96.29 icmp\_seq=14 Destination Port Unreachable

From 200.200.96.29 icmp\_seq=15 Destination Port Unreachable

64 bytes from 200.200.96.29: icmp\_req=16 ttl=61 time=1.06 ms

From 200.200.96.29 icmp\_seq=17 Destination Port Unreachable

除了/minute外，还有/second /hour /day可以用。

## 扩展模块icmp

icmp模块，可以使用--icmp-type设置报文的type，例如，我想主机PING出去能通，但是拒绝别人PING自己，可以：

do ~ # iptables -A INPUT -p icmp --icmp-type "echo-request" -j REJECT

do ~ # iptables --line-number -L -nv

Chain INPUT (policy ACCEPT 338 packets, 71847 bytes)

num pkts bytes target prot opt in out source destination

1 0 0 REJECT icmp -- \* \* 0.0.0.0/0 0.0.0.0/0 icmp type 8 reject-with icmp-port-unreachable

## 扩展模块state

如果想实现，我可以访问别人，别人不可以访问我，可以直接禁止INPUT的某些dport，例如：

do ~ # iptables -A INPUT ! -s 127.0.0.0/24 -p tcp -m tcp --dport 80 -j REJECT

do ~ # iptables -L -nv --line-number

Chain INPUT (policy ACCEPT 12970 packets, 4276K bytes)

num pkts bytes target prot opt in out source destination

1 3 176 REJECT tcp -- \* \* !127.0.0.0/24 0.0.0.0/0 tcp dpt:80 reject-with icmp-port-unreachable

外部再也不能访问我的80端口，但是我可以访问别人的80端口。

还有一种实现，可以用连接状态state，

do ~ # iptables -t filter -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT # 已有链接属于ESTABLISHED

do ~ # iptables -t filter -A INPUT ! -s 127.0.0.0/24 -j REJECT #记得把本地地址排除，不然127.0.0.1也连接不上

do ~ # iptables -L -nv --line-number

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)

num pkts bytes target prot opt in out source destination

1 63 5657 ACCEPT all -- \* \* 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED

2 320 99576 REJECT all -- \* \* !127.0.0.0/24 0.0.0.0/0 reject-with icmp-port-unreachable

state(可针对所有类型数据tcp,udp,icmp)的状态有NEW, ESTABLISHED, RELATED, INVALID, UNTRACKED，其中，NEW是第一个包，回复了ACK变为ESTABLISHED

看我们是在INPUT上接受ESTABLISHED，因为自己主动发起的链接，第一个是在OUTPUT上的NEW，可以通过，而回复的变是ESTABLISHED。

反过来，如果是外部的首链接，INPUT处应该是NEW，被拒绝。

从而实现本地可以访问外面，外面不能访问本地的作用。

## 记录日志

iptables有一个动作叫LOG，其可以记录日志到syslog，首先应该配置rsyslog.conf，为日志单独选择文件，centos 6.9 默认是kernel输出到/var/log/message

如下，我配置三条规则，中间一条是LOG

do ~ # iptables -t filter -A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT

do ~ # iptables -t filter -A INPUT -p tcp -m state --state NEW -j LOG

do ~ # iptables -t filter -A INPUT -p tcp --dport 80 -m state --state NEW -j REJECT

do ~ # iptables -L

Chain INPUT (policy ACCEPT)

target prot opt source destination

ACCEPT tcp -- anywhere anywhere tcp dpt:ssh state NEW

LOG tcp -- anywhere anywhere state NEW LOG level warning

REJECT tcp -- anywhere anywhere tcp dpt:http state NEW reject-with icmp-port-unreachable

当从其他机器分别telnet本机22和80端口时，可以发现，80端口的有日志记录，22的没有，因为，22提前ACCEPT，根本不会走到第二条规则：

Jul 14 19:21:07 do kernel: IN=eth2 OUT= MAC=fc:aa:14:53:b4:b1:54:be:f7:0d:86:5b:08:00 SRC=200.200.96.28 DST=200.200.96.29 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=26436 DF PROTO=TCP SPT=62849 DPT=80 WINDOW=8192 RES=0x00 SYN URGP=0

## 表NAT

NAT，包括SNAT，DNAT，SNAT我没用的非常常见，一般走路由出去就需要设置，DNAT的场景一般是外网访问内外时需要。

例如，我在本机开放一个端口30000，让其重定向到bridge的docker ip 172.17.0.2 端口 5201可以：

iptables -t nat -A PREROUTING ! -i docker0 -p tcp --dport 30000 -j DNAT --to-destination 172.17.0.2:5201

注意：DNAT是在PREROUTING的时候生效。

当然，SNAT需要指定具体的IP地址，不是很方便，MASQUERADE是SNAT的一个很好的弥补，不需要指定具体translate的IP，会自动获取出网口IP，例如，桥接本机的docker0网口，其网段是172.34.0.0/16，我们可以指定，其走非docker0网口出去的时候，记得做一下SNAT，可以显示指定出网口IP：

iptables -t nat -A POSTROUTING -s 172.34.0.0/16 -j SNAT --to-source 200.200.96.29

也可以隐式使用出网口IP(这种情况注意指定-o)：

iptables -t nat -A POSTROUTING -s 172.34.0.0/16 ! -o docker0 -j MASQUERADE

do ~ # iptables -L

Chain POSTROUTING (policy ACCEPT 1 packets, 60 bytes)

num pkts bytes target prot opt in out source destination

1 11 748 MASQUERADE all -- \* !docker0 172.34.0.0/16 0.0.0.0/0

注意：SNAT/MASQUERADE是在POSTROUTING的时候生效。

还有一个端口REDIRECT的命令，可以将访问一个端口的REDIRECT到另外一个端口，例如：

do ~ # iptables -t nat -A PREROUTING -p tcp --dport 8080 -j REDIRECT --to-ports 80

do ~ # iptables -L -nv --line-number -t nat

Chain PREROUTING (policy ACCEPT 87 packets, 10405 bytes)

num pkts bytes target prot opt in out source destination

2 0 0 REDIRECT tcp -- \* \* 0.0.0.0/0 0.0.0.0/0 tcp dpt:8080 redir ports 80

这时候，访问本机的8080，其实会被重定向到本机的80端口。

#iptables -t nat -A PREROUTING ! -i docker0 -p tcp --dport 2023 -j DNAT --to-destination 172.17.0.3:22

## 表RAW

raw表，用于跟踪最原始的数据。示例如何使用记录iptables规则访问日志。

创建目录：

node-4718301371424 ~ # rm -rf /var/log/ulogd/; mkdir -p /var/log/ulogd/

启动日志进程：

node-4718301371424 ~ # /etc/init.d/ulogd start

注意查看变化：

node-4718301371424 ~ # lsmod | grep log

nfnetlink\_log 17892 1

nfnetlink 14696 4 nf\_tables,nfnetlink\_log,nfnetlink\_queue

node-4718301371424 ~ # cat /proc/net/netfilter/nf\_log

0 NONE (nfnetlink\_log)

1 NONE (nfnetlink\_log)

2 nfnetlink\_log (nfnetlink\_log)

3 NONE (nfnetlink\_log)

4 NONE (nfnetlink\_log)

5 NONE (nfnetlink\_log)

6 NONE (nfnetlink\_log)

7 nfnetlink\_log (nfnetlink\_log)

8 NONE (nfnetlink\_log)

9 NONE (nfnetlink\_log)

10 nfnetlink\_log (nfnetlink\_log)

11 NONE (nfnetlink\_log)

12 NONE (nfnetlink\_log)

也可以自行添加，例如2是IPv4：

node-4718301371424 ~ # echo "nfnetlink\_log" > /proc/sys/net/netfilter/nf\_log/2

我要TRACE所有到本机31633端口的tcp链接：

node-4718301371424 ~ # iptables -t raw -A PREROUTING -p tcp --dport 31633 -j TRACE

node-5792074976342 ~ # iptables -t raw -L -nv

Chain PREROUTING (policy ACCEPT 63266 packets, 28M bytes)

pkts bytes target prot opt in out source destination

24 1352 TRACE tcp -- \* \* 0.0.0.0/0 0.0.0.0/0 tcp dpt:31633

Chain OUTPUT (policy ACCEPT 62962 packets, 26M bytes)

pkts bytes target prot opt in out source destination

从其他机器访问本机(因为挂的是PREROUTING)，便会有日志：

node-5792074976342 ~ # tail –f /var/log/ulogd/ulogd\_syslogemu.log

Jul 21 17:22:30 node-4718301371424 TRACE: raw:PREROUTING:policy:2 IN=eth1 OUT= MAC=fe:fc:fe:0f:dc:a3:fe:fc:fe:a7:6b:06:08:00 SRC=172.100.158.172 DST=172.100.158.174 LEN=60 TOS=00 PREC=0x00 TTL=64 ID=8241 DF PROTO=TCP SPT=46204 DPT=31633 SEQ=3349349723 ACK=0 WINDOW=29200 SYN URGP=0 MARK=0

Jul 21 17:22:30 node-4718301371424 TRACE: nat:PREROUTING:rule:1 IN=eth1 OUT= MAC=fe:fc:fe:0f:dc:a3:fe:fc:fe:a7:6b:06:08:00 SRC=172.100.158.172 DST=172.100.158.174 LEN=60 TOS=00 PREC=0x00 TTL=64 ID=8241 DF PROTO=TCP SPT=46204 DPT=31633 SEQ=3349349723 ACK=0 WINDOW=29200 SYN URGP=0 MARK=0

## 自定义链

创建自定义链的目的是为了方便管理，就像编程里面有了语句，也希望有函数，通过一个函数名，可以引用函数内的多条语句。

创建链：

do ~ # iptables -t filter -N MYLINK

向链中添加规则：

do ~ # iptables -A MYLINK ! -s 127.0.0.0/24 -p tcp -m tcp --dport 80 -j REJECT

do ~ # iptables -L

Chain MYLINK (0 references)

target prot opt source destination

REJECT tcp -- !loopback/24 anywhere tcp dpt:http reject-with icmp-port-unreachable

目前链已经创建，但是还没有被默认链引用，不会生效，我们让其在INPUT链上生效，例如，让tcp协议走MYLINK：

do ~ # iptables -t filter -A INPUT -p tcp -m tcp -j MYLINK

do ~ # iptables -L

Chain INPUT (policy ACCEPT)

target prot opt source destination

MYLINK tcp -- anywhere anywhere tcp

Chain MYLINK (1 references)

target prot opt source destination

REJECT tcp -- !loopback/24 anywhere tcp dpt:http reject-with icmp-port-unreachable

删除自定义链：

do ~ # iptables -t filter -X MYLINK # 不能删除，是因为INPUT链上还引用着它

iptables: Too many links.

do ~ # iptables -t filter -D INPUT 1 # 删除INPUT链上的引用

do ~ # iptables -t filter -X MYLINK # 再次不能删除，是因为MYLINK内部有内容，需要先清空

iptables: Directory not empty.

do ~ # iptables -t filter -F MYLINK # 清空MYLINK内部内容

do ~ # iptables -t filter -X MYLINK # 再次删除成功

## 规则备份和恢复

iptables-save 用于dump当前的所有IPTABLES规则，指定-t可以单独dump某表，dump默认到stdout，可以重定向到/etc/sysconfig/iptables，以便启动时候自动使用iptables-restore(该命令默认从stdin读取规则)加载此文件。

2017/7/17