iptables规则管理原则

1. iptables每条链都有默认的策略，意思是如果报文和指定的规则没有匹配，则使用默认的策略。当默认策略是DROP时，则指定的规则是以白名单的形式存在的。
2. 在CentOS7中默认使用firewalld前端规则管理工具，如果需要使用iptables以服务的形式运行管理，需要下载安装yum install iptables-services。
3. 添加规则时，首先判断所添加规则的功能，是进行过滤还是转发，然后判断在报文流经的那个位置进行匹配，最后判断得出规则索要添加的链的名称。
4. 同类型的规则，匹配范围小的方式链的前面。
5. 使用-j选项指定操作动作(target)
6. 查看iptables的扩展：rpm -ql iptables | grep "[[:lower:]]+.so$"，使用-m选项指明使用扩展模块

iptables命令使用格式

查看规则

```
# -t 指定要查看的链，不使用-t指定，则默认是filter表
# -n 以数字形式显示地址和端口
# -v 显示详细信息
# --line-numbers 显示规则编号
# -x 精确值显示计数器结果
iptables -t TABLE -n -v --line-numbers -x
```

链的管理

```
# 清空指定表上的所有规则
iptables -t TABLE -F
# 在指定表上创建一个自定义链
iptables -t TABLE -N
# 删除指定表上非内置的链，且链上不得有任何规则，必须为空链，不指定链名称，则删除所有自定义链
iptables -t TABLE -X CHAIN_NAME
# 清零指定表上规则的计数器，如果不指定链，则清零所有规则的计数器
iptables -t TABLE -Z CHAIN_NAME
# 修改指定表上指定链的默认策略
iptables -t TABLE -P CHAIN_NAME NEW_CHAIN_POLICY
# 修改自定义链的名称
iptables -t TABLE -E OLD_CHAIN_NAME NEW_CHAIN_NAME
```

规则管理

```
# 将新的规则追加于指定链的尾部
iptables -t TABLE -A RULE-SPECIFICATION
# 替换指定表中指定链的某条规则
iptables -t TABLE -R CHAIN_NAME RULE_NUMBER RULE_SPECIFICATION
# 将新的规则插入指定链的指定位置
iptables -t TABLE -I CHAIN_NAME RULE_NUMBER RULE_SPECIFICATION
# 删除链上的指定规则
iptables -t TABLE -D CHAIN_NAME RULE_NUMBER
```

iptables规则匹配

基本匹配

> 语法格式

```
#源IP地址匹配,可以省略，省略默认是所有IP地址
[!] -s|--src|--source IP|NETWORK
#目的IP地址匹配，可以省略，省略默认是所有IP地址
[!] -d|--dst|--dest|--destination IP|NETWORK
#检查报文中的协议，即IP首部中的protocols所标识的协议
-p|--protlcol {tcp|udp|icmp}
#数据报文的流入接口，仅能用于PREROUTING,INPUT,FORWARD链
-i|--in-interface INTERFACE_NAME
#数据报文的流出接口，仅能用于FORWRAD,OUTPUT,POSTROUTING链
-o|--out-interface INTERFACE_NAME
```

> 示例

```
# 放行所有从本机流出及流入本机的tcp报文
iptables -t filter -A  OUTPUT -s 192.168.1.150 -d 0.0.0.0/24 -p tcp -j ACCEPT
iptables -t filter -A INPUT -d 192.168.1.150 -s 0.0.0.0/24 -p tcp -j ACCEPT
#从网卡接口的位置报文
iptables -t filter -A OUTPUT -s 192.168.1.150 -o eth0 -j ACCEPT
iptables -t filter -A INPUT -d 192.168.1.150 -i eth0 -j ACCEPT
```

扩展匹配

> *报文类型及端口匹配*

```
#语法格式
# 目标端口，可以是单个端口，也可以是连续的多个端口。例如22-25
-p tcp --dport PORT
#源端口
-p tcp --sport PORT
#SYN | ACK | FIN | RST | PSH | URG ,检测6个标志位那些必须为1，那些必须为0
#检查在LIST1中所指明的所有标志位，且这其中LIST2中所表示的标志位必须为1，而余下的必须要为0 ， 没在LIST1中指明，不做检查。例如：--tcp-flages SYN,ACK,FIN,RST SYN
--tcp-flages FLAGE_LIST1 FLAGE_LIST2
#udp报文匹配
-p udp --dport PORT
-p udp --sport PORT
#icmp报文匹配
-p icmp --icmp-type 0|8
```

```
#示例
#放行22号端口，允许ssh服务连接
iptables -t filter -A INPUT -d 192.168.1.150 -p tcp --dport 22 -j ACCEPT
iptabess -t filter -A OUTPUT -s 192.168.1.150 -p tcp --sport 22 -j ACCEPT
#放行80端口，允许httpd服务连接
iptables -t filter -A INPUT -d 192.168.1.150 -p tcp --dport 80 -j ACCEPT
iptables -t filter -A OUTPUT -s 192.168.1.150 -p tcp --sport 80 -j ACCEPT
#不允许其他主机ping通本主机，允许本主机ping其他主机
iptables -t filter -A INPUT -d 192.168.1.150 -p icmp --icmp-type 0 -j ACCEPT
iptables -t filter -A OUTPUT -s 192.168.1.150 -p icmp --icmp-type 8 -j ACCEPT
```

> *multiport：以离散方式定义多端口匹配，最多定义15个端口*

```
#语法格式
#指明源端口
[!] --source-ports|--sports port1,port2,....
#指明目标端口
[!] --destination-ports|--dports port1,port2,....
```

```
#示例
#同时放行22，80端口
iptables -t filter -I INPUT -d 192.168.1.150 -p tcp -m multiport --dports 22,80 -j ACCEPT
iptables -t filter -I OUTPUT -s 192.168.1.150 -p tcp -m multiport --sports 22,80 -j ACCEPT
```

> *iprange：指明连续的IP地址范围时有效*

```
#语法格式
# 指明连续的源IP地址范围
[!] --src-range IP-IP
# 指明连续的目标IP地址范围
[!] --dst-range IP-IP
```

```
#示例
#给指定连续的IP放行22,80端口
iptabels -t filter -I INPUT -d 192.168.1.150 -p tcp -m multiport --dports 22,80 -m iprange --src-range 192.168.1.1-192.168.1.200 -j ACCEPT
iptables -t filer -I OUTPUT -s 192.168.1.150 -p tcp -m multiport --sport 22,80 -m iprange --dst-range 192.168.1.1-192.168.1.200 -j ACCEPT
```

> *string：匹配传输内容报文中出现的字符串*

```
#语法格式
#指明匹配算法
--algo bm|kmp
```

```
#示例
#拒绝报文中出现的“xue”字符串
iptables -t filter -I OUTPUT -m string --algo bm --string "xue" -j REJECT
#对通过80端口的请求报文进行匹配，并记录日志
iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string 'GET /index.html' -j LOG
```

> *connlimit：单IP并发访问数量限制*

```
#语法格式
# 小于数量N
--connlimit-upto N
#大于数量N
--connlimit-above N
```

```
#示例
#限制ssh服务的连接数
iptables -t filter -I INPUT -p tcp --dport 22 -m connlimit --connlimit-above 3 -j REJECT
```

> *state：根据连接追踪机制的状态，进而控制连接。*
>
> 连接追踪机制：iptables可以根据连接的状态制定出更加严密的防火墙规则。连接的状态有四种，分别是NEW，ESTABLISHED，RELATED和INVALID。iptables的扩展模块state可以实现针对连接状态制定防火墙规则。
>
> 连接追踪的记录都记录在/proc/net/nf_conntrack文件内，追踪记录最大记录条目数量/proc/sys/net/netfilter/nf_conntrack_max，需要注意的是追踪记录是保存在内核内存空间中的，所以当服务器请求量非常大时，需要调整追踪记录最大记录条目数，修改/etc/sysctl.conf增加配置项net.netfilter.nf_conntrack_max = N，net.nf_conntrack_max = N 。
>
> **连接追踪的四种状态：**
>
> **NEW** 连接追踪表中没有连接记录，即表示新发出的请求。

> **ESTABLSHED** 连接追踪表中有连接的记录，表示已经建立连接，每一个连接都有一个有效期，在未失效之前，连接状态都是ESTABKSHED，如果失效，则下一次连接被认为的NEW连接。
>
> **RELATED** 相关连接。当一个连接与状态是ESTABLSHED的连接有关时的连接状态是RELATED。例如，ftp服务连接需要命令连接和数据传输连接，当命令连接建立且状态是ESTABLSHED，此时建立的数据传输连接状态就是RELATED。
>
> **INVALID** 无法识别的连接。

```
#示例
#限制通过80响应的报文只能是被动的
iptabels -t filter -A INPUT -d 192.168.1.150 -p tcp --dport 80 -m state --state NEW,ESTABLSHED -j ACCEPT
iptables -t filter -A OUTPUT -s 192.168.1.150 -p tcp --sport 80 -m state --state ESTABLSHED -j ACCEPT
#ESTABLISHED状态的连接通信报文都放行
iptables -I INPUT -m state --state ESTABLISHED -j ACCEPT
iptables -I OUTPUT -m state --state ESTABLISHED -j ACCEPT
```

> 利用追踪连接机制放行ftp服务。需要借助连接追踪状态的RELATED，对于21号端口对NEW、ESTABLISHED放行，因为命令连接是通过21号端口连接的，而数据连接通过随机端口连接。

```
#装载RELATED追踪专用模块/lib/modules/2.6.32-573.el6.x86_64/kernel/net/netfilter/nf_conntrack_ftp.ko
modprobe nf_conntrack_ftp
#放行请求报文
#21号端口，命令连接：NEW | ESTABLISHED
iptables -I INPUT -d 192.168.1.155 -p tcp --dport 21 -m state --state NEW,ESTABLISHED -j ACCEPT
#数据连接：RELATED | ESTABLISHED
iptables -I INPUT -d 192.168.1.155 -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
#放行响应报文
iptables -I OUTPUT -m state --state ESTABLISHED -j ACCEPT
```

iptables规则的保存及重载

> iptables的规则保存在/etc/sysconfig/iptables文件内，使用重定向的方式保存规则，例如：iptables-sve >> /etc/sysconfig/iptables。
>
> 从指定文件加载iptables规则：iptables-restore < /path/to/iptables-rule。
>
> iptables以服务的形式管理运行，但事实上iptables没有启动任何进程，重启停止服务是重载规则以及清空规则的一个过程。