NAT配置：

sudo iptables -t nat -A POSTROUTING -s 192.168.168.0/24 -o wlp3s0 -j MASQUERADE

sudo iptables -t nat -A POSTROUTING -s 192.168.168.0/255.255.255.0 -j SNAT --to-source 192.168.123.180

iptables -t nat -L POSTROUTING -n --line-numbers -v

iptables -t nat -D POSTROUTING 1

显示路由表：

netstat -nr

显示Iptables：

iptables -L -n –line-number

20条IPTables防火墙规则用法！

很多用户把 Linux 中的 IPTables 当成一个防火墙，从严格意见上来说 IPTables 只是能够帮助管理员定义各种规则并与 Linux Kernel 进行沟通的一个命令行工具。它只是帮助管理员配置网络流量的传入、传出规则列表，具体的实现其实是在 Linux 内核当中。

IPTables 包括一组内置和由用户定义规则的「链」，管理员可以在「链」上附加各种数据包处理规则。

FILTER 默认过滤表，内建的链有：

INPUT：处理流入本地的数据包

FORWARD：处理通过系统路由的数据包

OUTPUT：处理本地流出的数据包

NAT 实现网络地址转换的表，内建的链有：

PREROUTING：处理即将接收的数据包

OUTPUT：处理本地产生的数据包

POSTROUTING：处理即将传出的数据包

MANGLE 此表用于改变数据包，共 5 条链：

PREROUTING：处理传入连接

OUTPUT：处理本地生成的数据包

INPUT：处理报文

POSTROUTING：处理即将传出数据包

FORWARD：处理通过本机转发的数据包

1、启动、停止和重启IPTables

虽然 IPTables 并不是一项服务，但在 Linux 中还是可以像服务一样对其状态进行管理。

基于SystemD的系统

systemctl start iptablessystemctl stop iptablessystemctl restart iptables

基于SysVinit的系统

/etc/init.d/iptables start/etc/init.d/iptables stop/etc/init.d/iptables restart

2、查看IPtables防火墙策略

你可以使用如下命令来查看 IPtables 防火墙策略：

iptables -L -n -v

以上命令是查看默认的 FILTER 表，如果你只希望查看特定的表，可以在 -t 参数后跟上要单独查看的表名。例如只查看 NAT 表中的规则，可以使用如下命令：

iptables -t nat -L -v –n

3、屏蔽某个IP地址

如果你发布有某个 IP 向服务器导入攻击或非正常流量，可以使用如下规则屏蔽其 IP 地址：

iptables -A INPUT -s xxx.xxx.xxx.xxx -j DROP

注意需要将上述的 XXX 改成要屏蔽的实际 IP 地址，其中的 -A 参数表示在 INPUT 链的最后追加本条规则。（IPTables 中的规则是从上到下匹配的，一旦匹配成功就不再继续往下匹配）

如果你只想屏蔽 TCP 流量，可以使用 -p 参数的指定协议，例如：

iptables -A INPUT -p tcp -s xxx.xxx.xxx.xxx -j DROP

4、解封某个IP地址

要解封对 IP 地址的屏蔽，可以使用如下命令进行删除：

iptables -D INPUT -s xxx.xxx.xxx.xxx -j DROP

其中 -D 参数表示从链中删除一条或多条规则。

5、使用IPtables关闭特定端口

很多时候，我们需要阻止某个特定端口的网络连接，可以使用 IPtables 关闭特定端口。

阻止特定的传出连接：

iptables -A OUTPUT -p tcp --dport xxx -j DROP

阻止特定的传入连接：

iptables -A INPUT -p tcp --dport xxx -j ACCEPT

6、使用Multiport控制多端口

使用 multiport 我们可以一次性在单条规则中写入多个端口，例如：

iptables -A INPUT -p tcp -m multiport --dports 22,80,443 -j ACCEPTiptables -A OUTPUT -p tcp -m multiport --sports 22,80,443 -j ACCEPT

7、在规则中使用 IP 地址范围

在 IPtables 中 IP 地址范围是可以直接使用 CIDR 进行表示的，例如：

iptables -A OUTPUT -p tcp -d 192.168.100.0/24 --dport 22 -j ACCEPT

8、配置端口转发

有时我们需要将 Linux 服务器的某个服务流量转发到另一端口，此时可以使用如下命令：

iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 25 -j REDIRECT --to-port 2525

上述命令会将所有到达 eth0 网卡 25 端口的流量重定向转发到 2525 端口。

9、屏蔽HTTP服务Flood攻击

有时会有用户在某个服务，例如 HTTP 80 上发起大量连接请求，此时我们可以启用如下规则：

iptables -A INPUT -p tcp --dport 80 -m limit --limit 100/minute --limit-burst 200 -j ACCEPT

上述命令会将连接限制到每分钟 100 个，上限设定为 200。

10、禁止PING

对 Linux 禁 PING 可以使用如下规则屏蔽 ICMP 传入连接：

iptables -A INPUT -p icmp -i eth0 -j DROP

11、允许访问回环网卡

环回访问（127.0.0.1）是比较重要的，建议大家都开放：

iptables -A INPUT -i lo -j ACCEPTiptables -A OUTPUT -o lo -j ACCEPT

12、屏蔽指定MAC地址

使用如下规则可以屏蔽指定的 MAC 地址：

iptables -A INPUT -m mac --mac-source 00:00:00:00:00:00 -j DROP

13、限制并发连接数

如果你不希望来自特定端口的过多并发连接，可以使用如下规则：

iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 3 -j REJECT

以上规则限制每客户端不超过 3 个连接。

14、清空IPtables规则

要清空 IPtables 链可以使用如下命令：

iptables -F

要清空特定的表可以使用 -t 参数进行指定，例如：

iptables -t nat –F

15、保存IPtables规则

默认情况下，管理员对 IPtables 规则的操作会立即生效。但由于规则都是保存在内存当中的，所以重启系统会造成配置丢失，要永久保存 IPtables 规则可以使用 iptables-save 命令：

iptables-save > ~/iptables.rules

保存的名称大家可以自己改。

16、还原IPtables规则

有保存自然就对应有还原，大家可以使用 iptables-restore 命令还原已保存的规则：

iptables-restore < ~/iptables.rules

17、允许建立相关连接

随着网络流量的进出分离，要允许建立传入相关连接，可以使用如下规则：

iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

允许建立传出相关连接的规则：

iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT

18、丢弃无效数据包

很多网络攻击都会尝试用黑客自定义的非法数据包进行尝试，我们可以使用如下命令来丢弃无效数据包：

iptables -A INPUT -m conntrack --ctstate INVALID -j DROP

19、IPtables屏蔽邮件发送规则

如果你的系统不会用于邮件发送，我们可以在规则中屏蔽 SMTP 传出端口：

iptables -A OUTPUT -p tcp --dports 25,465,587 -j REJECT

20、阻止连接到某块网卡

如果你的系统有多块网卡，我们可以限制 IP 范围访问某块网卡：

iptables -A INPUT -i eth0 -s xxx.xxx.xxx.xxx -j DROP

源地址可以是 IP 或 CIDR。

iptables超全详解

前提基础：

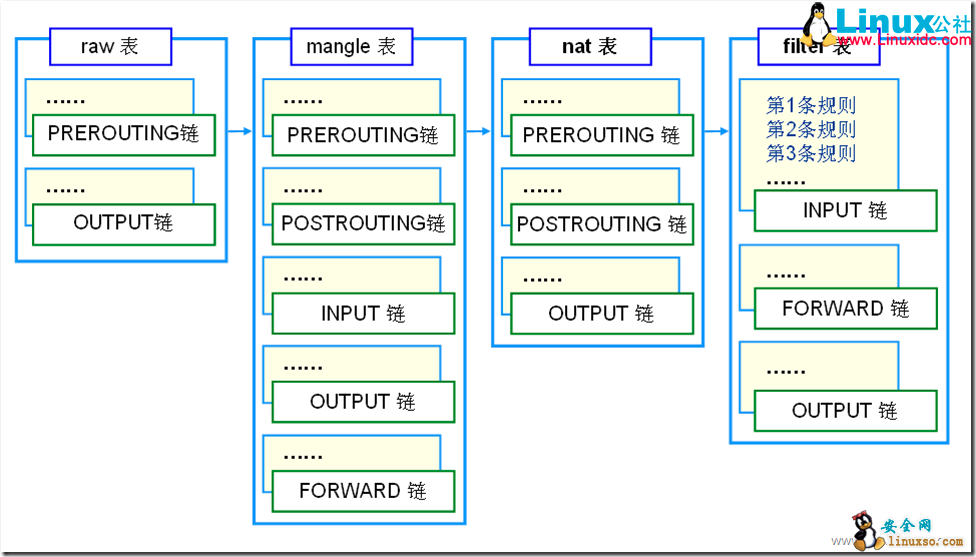
当主机收到一个数据包后，数据包先在内核空间中处理，若发现目的地址是自身，则传到用户空间中交给对应的应用程序处理，若发现目的不是自身，则会将包丢弃或进行转发。

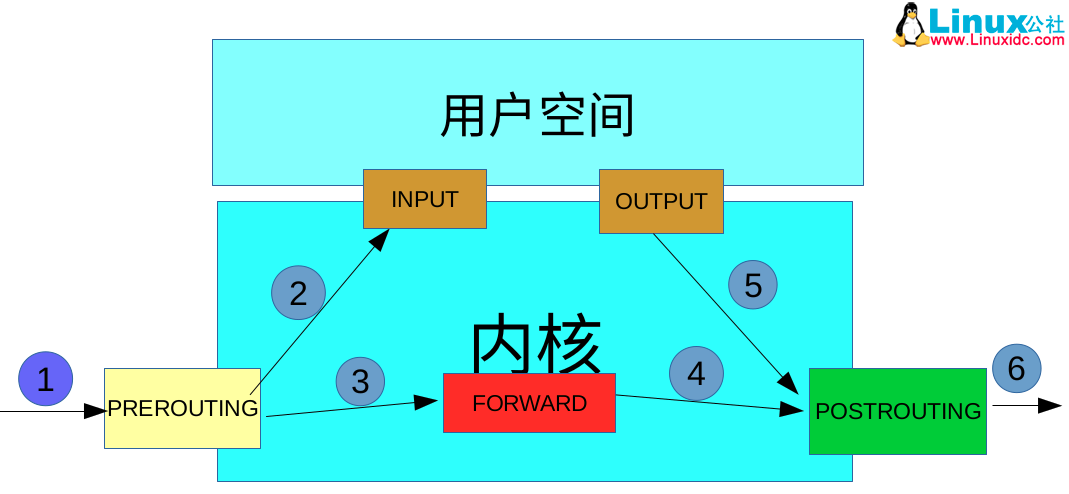
iptables实现防火墙功能的原理是：在数据包经过内核的过程中有五处关键地方，分别是PREROUTING、INPUT、OUTPUT、FORWARD、POSTROUTING，称为钩子函数，iptables这款用户空间的软件可以在这5处地方写规则，对经过的数据包进行处理，规则一般的定义为“如果数据包头符合这样的条件，就这样处理数据包”。

iptables中定义有5条链，说白了就是上面说的5个钩子函数，因为每个钩子函数中可以定义多条规则，每当数据包到达一个钩子函数时，iptables就会从钩子函数中第一条规则开始检查，看该数据包是否满足规则所定义的条件。如果满足，系统就会根据该条规则所定义的方法处理该数据包；否则iptables将继续检查下一条规则，如果该数据包不符合钩子函数中任一条规则，iptables就会根据该函数预先定义的默认策略来处理数据包

iptables中定义有表，分别表示提供的功能，有filter表（实现包过滤）、nat表（实现网络地址转换）、mangle表（实现包修改）、raw表（实现数据跟踪），这些表具有一定的优先级：raw-->mangle-->nat-->filter

一条链上可定义不同功能的规则，检查数据包时将根据上面的优先级顺序检查





数据包先经过PREOUTING，由该链确定数据包的走向：

    1、目的地址是本地，则发送到INPUT，让INPUT决定是否接收下来送到用户空间，流程为①--->②;

    2、若满足PREROUTING的nat表上的转发规则，则发送给FORWARD，然后再经过POSTROUTING发送出去，流程为： ①--->③--->④--->⑥

主机发送数据包时，流程则是⑤--->⑥

iptables安装配置

linux一般默认都已经安装iptables，只需要开启服务即可

|  |  |
| --- | --- |
| 1 | service iptables start |

iptables规则书写

基本语法：iptables [-t 表] [操作命令] [链][规则匹配器][-j 目标动作]

|  |  |  |
| --- | --- | --- |
| 表 | 说明 | 支持的链 |
| raw | 一般是为了不再让iptables对数据包进行跟踪，提高性能 | PREROUTING、OUTPUT |
| mangle | 对数据包进行修改 | 五个链都可以 |
| nat | 进行地址转换 | PREROUTING、OUTPUT、POSTROUTING |
| filter（默认） | 对包进行过滤 | INPUT、FORWARD、OUTPUT |

|  |  |
| --- | --- |
| 常用操作命令 | 说明 |
| -A | 在指定链尾部添加规则 |
| -D | 删除匹配的规则 |
| -R | 替换匹配的规则 |
| -I | 在指定位置插入规则  例：iptables -I INPUT 1 --dport 80 -j ACCEPT  （将规则插入到filter表INPUT链中的第一位上） |
| -L/S | 列出指定链或所有链的规则 |
| -F | 删除指定链或所有链的规则 |
| -N | 创建用户自定义链  例：iptables -N allowed |
| -X | 删除指定的用户自定义链 |
| -P | 为指定链设置默认规则策略，对自定义链不起作  用  例：iptables -P OUTPUT DROP |
| -Z | 将指定链或所有链的计数器清零 |
| -E | 更改自定义链的名称  例：iptables -E allowed disallowed |
| -n | ip地址和端口号以数字方式显示  例：iptables -Ln |

|  |  |
| --- | --- |
| 常见规则匹配器 | 说明 |
| -p tcp|udp|icmp|all | 匹配协议，all会匹配所有协议 |
| -s addr[/mask] | 匹配源地址 |
| -d addr[/mask] | 匹配目标地址 |
| --sport port1[:port2] | 匹配源端口(可指定连续的端口） |
| --dport port1[:port2] | 匹配目的端口(可指定连续的端口） |
| -o interface | 匹配出口网卡，只适用FORWARD、POSTROUTING、OUTPUT。  例：iptables -A FORWARD -o eth0 |
| -i interface | 匹配入口网卡，只使用PREROUTING、INPUT、FORWARD。 |
| --icmp-type | 匹配icmp类型（使用iptables -p icmp -h可查看可用的ICMP类型） |
| --tcp-flags mask comp | 匹配TCP标记，mask表示检查范围，comp表示匹配mask中的哪些标记。  例：iptables -A FORWARD -p tcp --tcp-flags ALL SYN，ACK -j ACCEPT  （表示匹配SYN和ACK标记的数据包） |

|  |  |
| --- | --- |
| 目标动作 | 说明 |
| ACCEPT | 允许数据包通过 |
| DROP | 丢弃数据包 |
| REJECT | 丢弃数据包，并且将拒绝信息发送给发送方 |
| SNAT | 源地址转换（在nat表上）  例：iptables -t nat -A POSTROUTING -d 192.168.0.102 -j SNAT --to 192.168.0.1 |
| DNAT | 目标地址转换（在nat表上）  例：iptables -t nat -A PREROUTING -d 202.202.202.2 -j DNAT --to-destination 192.168.0.102 |
| REDIRECT | 目标端口转换（在nat表上）  例：iptables -t nat -D PREROUTING -p tcp --dport 8080 -i eth2.2 -j REDIRECT --to 80 |
| MARK | 将数据包打上标记  例：iptables -t mangle -A PREROUTING -s 192.168.1.3 -j MARK --set-mark 60 |

注意要点：

    1、目标地址转换一般在PREROUTING链上操作

    2、源地址转换一般在POSTROUTING链上操作

保存和恢复iptables规则

    使用iptables-save可以保存到特定文件中

|  |  |
| --- | --- |
| 1 | iptables-save >/etc/sysconfig/iptables\_save |

    使用iptables-restore可以恢复规则

|  |  |
| --- | --- |
| 1 | iptables-restore</etc/sysconfig/iptables\_save |

iptables的进阶使用

   1、limit限制流量：

        -m limit --limit-burst 15        #设置一开始匹配的最大数据包数���

        -m limit --limit 1000/s            #设置最大平均匹配速率

        -m limit --limit 5/m --limit-burst 15     #表示一开始能匹配的数据包数量为15个，每匹配到一个，

                                                                    limit-burst的值减1,所以匹配到15个时，该值为0,以后每过

                                                                    12s，limit-burst的值会加1,表示又能匹配1个数据包

例子：

|  |  |
| --- | --- |
| 1  2 | iptables -A INPUT -i eth0 -m limit --limit 5/m --limit-burst 15 -j ACCEPT  iptables -A INPUT -i eth0 -j DROP |

    注意要点：

        1、--limit-burst的值要比--limit的大

        2、limit本身没有丢弃数据包的功能，因此，需要第二条规则一起才能实现限速的功能

    2、time ：在特定时间内匹配

|  |  |
| --- | --- |
| -m time | 说明 |
| --monthdays day1[,day2] | 在每个月的特定天匹配 |
| --timestart hh:mm:ss | 在每天的指定时间开始匹配 |
| --timestop hh:mm:ss | 在每天的指定时间停止匹配 |
| --weekdays day1[,day2] | 在每个星期的指定工作日匹配，值可以是1-7 |

例子：

|  |  |
| --- | --- |
| 1  2 | iptables -A INPUT -i eth0 -m time --weekdays 1,2,3,4 -jACCEPT  iptables -A INPUT -i eth0 -j DROP |

    3、ttl：匹配符合规则的ttl值的数据包

|  |  |
| --- | --- |
| 参数 | 说明 |
| --ttl-eq 100 | 匹配TTL值为100的数据包 |
| --ttl-gt 100 | 匹配TTL值大于100的数据包 |
| --ttl-lt 100 | 匹配TTL值小于100的数据包 |

例子：

|  |  |
| --- | --- |
| 1 | iptables -A OUTPUT -m ttl --ttl-eq 100 -j ACCEPT |

    4、multiport：匹配离散的多个端口

|  |  |
| --- | --- |
| 参数 | 说明 |
| --sports port1[,port2,port3] | 匹配源端口 |
| --dports port1[,port2,port3] | 匹配目的端口 |
| --ports port1[,port2,port3] | 匹配源端口或目的端口 |

例子：

|  |  |
| --- | --- |
| 1 | iptables -A INPUT -m multiport --sports 22，80，8080 -j DROP |

    5、state：匹配指定的状态数据包

|  |  |
| --- | --- |
| 参数 | 说明 |
| --state value | value可以为NEW、RELATED（有关联的）、ESTABLISHED、INVALID（未知连接） |

例子：

|  |  |
| --- | --- |
| 1 | iptables -A INPUT -m state --state NEW，ESTABLISHED -j ACCEPT |

    6、mark：匹配带有指定mark值的数据包

|  |  |
| --- | --- |
| 参数 | 说明 |
| --mark value | 匹配mark标记为value的数据包 |

例子：

|  |  |
| --- | --- |
| 1 | iptables -t mangle -A INPUT -m mark --mark 1 -j DROP |

    7、mac：匹配特定的mac地址

例子：

|  |  |
| --- | --- |
| 1 | iptables -A FORWARD -m mac --mac-source 00:0C:24:FA:19:80 -j DROP |

基本概念：

1.防火墙工作在主机边缘:对于进出本网络或者本主机的数据报文，根据事先设定好的检查规则对其检查，对形迹可疑的报文一律按照事先定义好的处理机制做出相应处理

对linux而言tcp/ip协议栈是在内核当中，意味着报文的处理是在内核中处理的，也就是说防火墙必须在工作在内核中，防火墙必须在内核中完成tcp/ip报文所流进的位置，用规则去检查，才真正能工作起来。

iptables用来衡量tcp/ip报文的属性：源ip、目标ip、源端口、目标端口；

tcp标志位: syn、syn+ack、ack、 fin、urg、psh、rst ；

2.应用网关

众多代理服务器都是应用网关，比如squid（使用acl限制应用层）varish这一类代理服务等。

3，入侵检测系统（IDS）：

·网络入侵检测系统 NIDS

·主机入侵检测系统 HIDS

对于IDS常用的检测服务有：snort等

4.入侵防御系统（IPS），比如蜜罐

部署一套入侵检测系统是非常麻烦的，因为必须检测网络任意一个位置

对于IPS常用的检测服务有： tripwire 等

iptables基本概念

对linux来说，是能够实现主机防火墙的功能组件，如果部署在网络边缘，那么既可以扮演网络防火墙的角色，而且是纯软件的

网络数据走向：

请求报文à网关à路由à应用程序（等待用户请求）à内核处理à路由à发送报文

iptables规则功能

表:

filter主要和主机自身有关，主要负责防火墙功能 过滤本机流入流出的数据包是默认使用的表;

input :负责过滤所有目标地址是本机地址的数据包，就是过滤进入主机的数据包;

forward :负责转发流经主机但不进入本机的数据包，和NAT关系很大;

output :负责处理源地址的数据包，就是对本机发出的数据包;

NAT表：

负责网络地址转换，即来源于目的IP地址和端口的转换，一般用于共享上网或特殊端口的转换服务

snat :地址转换

dnat :标地址转换

pnat :标端口转换

mangle 表：

将报文拆开来并修改报文标志位，最后封装起来

5个检查点（内置链）

·PREROUTING

·INPUT

·FORWORD

·OUTPUT

·POSTROUTING

多条链整合起来叫做表，比如，在input这个链，既有magle的规则也可能有fileter的规则。因此在编写规则的时候应该先指定表，再指定链

netfilter主要工作在tcp/ip协议栈上的，主要集中在tcp报文首部和udp报文首部

规则的属性定义：

1.网络层协议

主要集中在ip协议报文上

2.传输层协议属性：

主要集中在

tcp

udp

icmp icmp其并不是真正意义传输层的，而是工作在网络层和传输层之间的一种特殊的协议

3.ip报文的属性：

IP报文的属性为: 源地址.目标地址

4.iptables规则匹配

iptables如何查看表和链

大写字母选项：可以实现某种功能，比如添加删除清空规则链；

小写字母选项：用来匹配及其他；

-L ：list 列表

-n :数字格式显示ip和端口；

--line-numbers:显示行号；

-x ： 显示精确值，不要做单位换算；

-t : 指定表

-t{fillter|nat|mangle|raw}

-v ： 显示详细信息 -v -vvv -vvvv ..可以显示更详细的信息

5.其他子命令：

管理链：

-F ：清空链

清空nat表中的input链，格式如下：

#iptables-t nat -F INPUT

#清空fllter表所有链：

#iptables-F

-P : 设定默认策略，为指定链设置默认策略，格式如下：

#设置fllter表input链的默认规则为丢弃

iptables-t fllter -P INPUT DROP

-N ： 新建一条自定义链（内置链不能删除，如果太多，可以自定义链）

#自定义连只能被调用才可以发挥作用

iptables-N fillter\_web

-X : 删除自定义空链，如果链内有规则，则无法删除

-Z ：计算器清零

iptables-Z

-E ：重命名自定义链

iptables管理规则：

-A ：append附加规则，将新增的规则添加到链的尾部

-I[n] ：插入为第n条规则

-D : 删除第n条规则

-R[n] : 替换第N条

表和链的对应关系：

fillter ：INPUT FORWORD OUTPUT

nat : PREROUTING POSTROUTING OUTPUT

使用-t指定表来查看指定表内的规则：

#iptables-t nat -L -n

raw : prerouting output

iptables-t raw -L -n

mangle: prerouting input forword output postrouting

iptables-t mangle -L -n

#查看规则

[root@test3~]# iptables -L -n

Chain INPUT (policy ACCEPT)

target prot opt source destination

Chain FORWARD (policy ACCEPT)

target prot optsource destination

Chain OUTPUT (policy ACCEPT)

target prot optsource destination

通过以上可以观察到，每一个链都有默认策略：policy ACCEPT

通常只需要修改fllter表的默认策略即可，由此如果有报文请求来访问本机的某个服务，那么则会经过input链，因此进来的报文都是需要做过滤的，那么出去的报文则不需要过滤，在有些特定的场所下也需要做过滤

所以写规则的时候必须放将规则写在正确链上，意义非常重大

规则和默认策略都有2个计数器，通过-v选项可以观察规则的匹配情况

#iptables -t nat -L -n -v

[root@sshgw~]# iptables -L -n -v

ChainINPUT (policy ACCEPT 7 packets, 975 bytes)

pkts bytestarget prot opt in out source destination

0 0 ACCEPT all -- lo \* 0.0.0.0/0 0.0.0.0/0

0 0 DROP all -- eth2 \* 101.61.0.0/10 0.0.0.0/0

0 0 DROP all -- eth2 \* 127.0.0.0/8 0.0.0.0/0

0 0 DROP all -- eth2 \* 162.254.0.0/16 0.0.0.0/0

0 0 DROP all -- eth2 \* 192.0.0.0/24 0.0.0.0/0

0 0 DROP all -- eth2 \* 192.0.2.0/24 0.0.0.0/0

0 0 DROP all -- eth2 \* 197.18.0.0/15 0.0.0.0/0

0 0 DROP all -- eth2 \* 197.51.100.0/24 0.0.0.0/0

0 0 DROP all -- eth2 \* 203.0.111.0/24 0.0.0.0/0

0 0 DROP all -- eth2 \* 224.0.0.0/4 0.0.0.0/0

0 0 DROP all -- eth2 \* 240.0.0.0/4 0.0.0.0/0

776 37056 REFRESH\_TEMP all -- \* \* 0.0.0.0/0 0.0.0.0/0

编写规则语法：

iptables [-t 表] 大写选项子命令 [规则号] 链名 匹配标准 -j 目标（规则）

目标：

DROP : 丢弃

REJECT : 拒绝

ACCEPT : 接受

RETURN ： 返回主链继续匹配

REDIRECT: 端口重定向

MASQUERADE :地址伪装

DNAT : 目标地址转换

SNAT ：源地址转换

MARK ：打标签

LOG

自定义链

匹配标准

iptables的匹配标准大致分为两类：

1.通用匹配

-s | --src | --source [!] IP/NETWORK

-d ------------------------

-i :指定数据报文流入接口 input prerouting forward

-o :指定数据报文流出接口 output postrouting forward

-p :明确说明只放行哪种协议的报文匹配规则

以当前主机为例：

凡是来自于某个ip段的网络访问本机

[root@test3xtables-1.4.7]# iptables -A INPUT -s 10.0.10.0/24 -d 10.0.10.0/24 -j ACCEPT

[root@test3 xtables-1.4.7]# iptables -L -n -v

ChainINPUT (policy ACCEPT 10 packets, 1029 bytes)

pkts bytestarget prot opt in out source destination

22 1660 ACCEPT all -- \* \* 10.0.10.0/24 10.0.10.0/24

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)

pkts bytes target prot opt in out source destination

Chain OUTPUT (policy ACCEPT 16 packets, 1536 bytes)

pkts bytes target prot opt in out source destination

pkts 被本机报文所匹配的个数

bytes 报文所有大小记起来之和

opt 额外的选项，--表示没有

target 处理机制

prot 放行哪种协议

source 源地址

destination 目标地址

对于严谨的规则，一般默认规则都是拒绝未知，允许已知

如下所示：

只放行信任IP地址段，其他全部禁止

iptables-P INPUT DORP

iptables-A INPUT -s 10.0.10.0/24 -d 10.0.10.0/24 -j ACCEPT

iptables-P OUTPUT DORP

iptables-A OUTPUT -d 10.0.10.0/24 -s 10.0.10.0/24-j ACCEPT

保存规则

[root@test3~]# /etc/init.d/iptables save

iptables:Saving firewall rules to /etc/sysconfig/iptables:[ OK ]

保存规则至其他文件

[root@test3~]# iptables-save > /tmp/iptables

加载iptables文件规则

[root@test3~]# iptables-resotre < /tmp/iptables

1.2.规则的替换

首先来查看规则

[root@test3 ~]# iptables -L -n --line-number

ChainINPUT (policy ACCEPT)

num target prot opt source destination

1 ACCEPT all -- 10.0.10.0/24 10.0.10.0/24

ChainFORWARD (policy DROP)

num target prot opt source destination

ChainOUTPUT (policy ACCEPT)

num target prot opt source destination

替换规则：将规则1替换为 eth0只能够通过某个网段进来

[root@test3~]# iptables -R INPUT 1 -s 10.0.10.0/24-d 10.0.10.62 -i eth0 -j ACCEPT

[root@test3~]# iptables -L -n --line-number

ChainINPUT (policy ACCEPT)

num target prot opt source destination

1 ACCEPT all -- 10.0.10.0/24 10.0.10.62

2.扩展匹配

#所有的扩展匹配表示要使用-m来指定扩展的名称来引用，而每个扩展模块一般都会有自己特有的专用选项，在这些选项中，有些是必备的：

2.1隐含扩展

如下所示：

#端口之间必须是连续的

-p tcp--sport|--dport 21-80

#取反，非21-80的端口

-p tcp--sport|--dport !21-80

#检测报文中的标志位

--tcp-flagsSYN,ACK,RST,FIN, SYN

ALL #表示为所有标志位

NONE #表示没有任何一个标志位

#--tcp-flags ALL NONE #表示所有标志位都检测，但是其中多有都为0

#--tcp-flage ALL SYN,FIN #表示SYN,FIN都为1（即握手又断开）

#生成环境下tcp-flags 用的非常多，意义非常重要

例：放行本机对web的访问

[root@test3~]# iptables -A INPUT -d 10.0.10.62 -ptcp --dport 80 -j ACCEPT

[root@test3~]# iptables -L -n

ChainINPUT (policy DROP)

target prot opt source destination

ACCEPT all -- 10.0.10.0/24 10.0.10.62

ACCEPT tcp -- 0.0.0.0/0 10.0.10.62 tcp dpt:80

放行出去的报文，源端口为80

[root@test3~]# iptables -A OUTPUT -s 10.0.10.62 -p tcp --sport 80 -j ACCEPT

查看匹配规则

[root@test3 ~]# iptables -L -n --line-number

ChainINPUT (policy DROP)

num target prot opt source destination

1 ACCEPT all -- 10.0.10.0/24 10.0.10.62

2 ACCEPT tcp -- 0.0.0.0/0 10.0.10.62 tcp dpt:80

ChainFORWARD (policy DROP)

num target prot opt source destination

ChainOUTPUT (policy DROP)

num target prot opt source destination

1 ACCEPT all -- 10.0.10.0/24 10.0.10.0/24

2 ACCEPT tcp -- 10.0.10.62 0.0.0.0/0 tcp spt:80

考虑要点：

（1）规则为放行出去的响应报文

（2）考虑源IP地址为本机，目标为访问的时候拆开报文才可以获知，而写规则的时候是面向所有主机，所以这里不用写

（3）源端口：80 ，因为用户访问的时候一定会访问其80端口，无可非议的

（4）目标端口：请求到来的时候事先无法断定对方的端口是多少，所以不用写

2.2协议匹配

通常对协议做匹配则使用 -p 参数 来指定协议即可

匹配UDP：UDP只有端口的匹配，没有任何可用扩展，格式如下

-p udp--sport | --dport

匹配ICMP格式如下

-picmp --icmp-[number]

icmp常见类型：请求为8（echo-request），响应为0(echo-reply)

例：默认规则input output 都为DROP,使其本机能ping（响应的报文）的报文出去

通过此机器去ping网关10.0.10.1 ， 可结果却提示not permitted，使其能通10.0.10.0/24网段中的所有主机

[root@test3~]#iptables -A OUTPUT -s 10.0.10.62 -d 10.0.10.0/24 -p icmp --icmp-type8 -j ACCEPT

可看到无法响应：0表示响应进来的报文规则，并没有放行自己作为服务端的的角色规则

[root@test3~]# iptables -A INPUT -s 10.0.10.0/24 -d 10.0.10.62 -p icmp --icmp-type0 -j ACCEPT

#ping 10.0.10.x

允许类型为0（响应报文）出去

[root@test3~]# iptables -A OUTPUT -s 10.0.10.62 -d 10.0.10.0/24 -picmp --icmp-type 0 -j ACCEPT

例2：本机DNS服务器，要为本地客户端做递归查询；iptables的input output默认为drop 本机地址是10.0.10.62

[root@test3~]# iptables -A INPUT -d 10.0.10.62 -p udp --dprot 53 -j ACCEPT

[root@test3~]# iptables -A OUTPUT -S 10.0.10.62 -p udp --sprot 53 -j ACCEPT

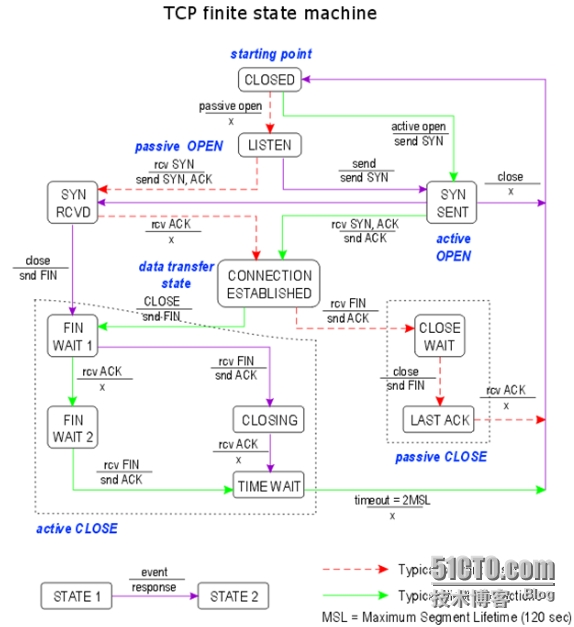
客户端请求可以进来，响应也可以出去，但是自己作为客户端请求别人是没有办法出去的，所以：

[root@test3~]# iptables -A OUTPUT -s 10.0.10.62 -p udp --dport 53 -j ACCEPT

[root@test3~]# iptables -A INPUT -d 10.0.10.62 -p udp --sprot 53 -j ACCEPT

如果为tcp 则将以上udp改为tcp即可

2.3 TCP协议的报文走向



TCP连接的建立

双方主机为了实现tcp的通信，所以首先三次握手

客户端主动发出了SYN，服务器端处于监听状态，随时等待客户端的请求信息；

服务器端接收到了SYN请求，从而回应用户的请求，发送SYN\_ACK ，从而转换为SYN\_REVIN

客户端在发出了请求，从发出的那一刻close状态转换为SYN\_SENT状态

客户端在SYN\_SENT状态中一旦收到了服务端发来的SYN\_ACK 之后，转换为ESTABLISHED状态，这时便可以开始传送数据了，无论怎么传都是ESTABLISHED状态

而服务器端收到了对方的ACK，同样处于ESTABLISHED状态

数据传输结束之后

客户端从ESTABLEISHED状态，发起四次断开请求

客户端发起FIN请求，从而进入等待状态

服务端收到断开请求之后，便发起ACK请求

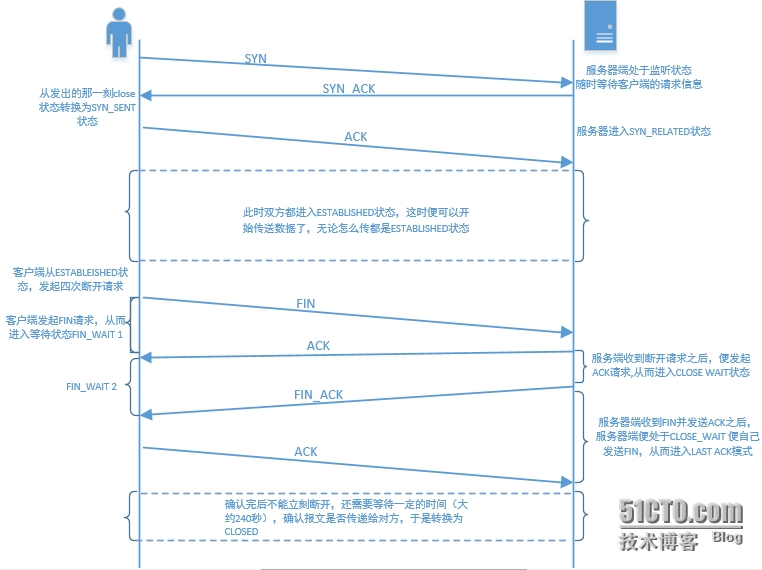
客户端收到服务端发来的ACK确认信息后，从而又发起FIN\_2 请求

等待服务端发来的FIN请求之后，便确认

服务器端收到FIN并发送ACK之后，服务器端便处于CLOSE\_WAIT便自己发送FIN，从而进入LAST ACK模式 ，

确认完后不能立刻断开，还需要等待一定的时间（大约240秒），确认报文是否传递给对方

于是转换为CLOSED



iptables中有一个扩张参数--status

此扩展可以追踪tcp udp icmp等各种状态

其能够使用某种内核数据结构保持此前曾经建立的连接状态时间的功能，称为连接追踪

内核参数文件路径为：

[root@test3~]# ls /proc/sys/net/netfilter/

[root@test3~]# cat /proc/sys/net/netfilter/nf\_conntrack\_udp\_timeout

30

以此为例，在其30秒钟内，曾经建立过的udp连接,这些连接都可以被追踪到的，可以明确知道在这期间哪个客户端曾经访问过，只要基于请求的序列，能跟此前保持会话信息，即可查询

2.4显式扩展

在iptalbes中数据包和被跟踪连接的4种不同状态相关联，这四种状态分别是NEW、ESTABLISHED、RELATED及INVALID，除了本机产生的数据包由NAT表的OUTPUT链处理外，所有连接跟踪都是在NAT表的PREROUTING链中进行处理的，也就是说iptables在NAT表的PREROUTING链里从新计算所有数据包的状态。如果发送一个流的初始化数据包，状态就会在NAT表的OUTPUT链里被设置为NEW，当收到回应的数据包时，状态就会在NAT表的PREROUTING链里被设置为ESTABLISHED，如果第一个数据包不是本机生成的，那就回在NAT表PREROUTING链里被设置为NEW状态，所以所有状态的改变和计算都是在NAT表中的表链和OUTPUT链里完成的。

使用-m来指定其状态并赋予匹配规则，语法如下

-mstate --state 状态

NEW

ESTABLISHED

RELATED

INVALID

NEW：

NEW状态的数据包说明这个数据包是收到的第一个数据包。比如收到一个SYN数据包，它是连接的第一个数据包，就会匹配NEW状态。第一个包也可能不是SYN包，但它仍会被认为是NEW状态。

ESTABLISHED：

只要发送并接到应答，一个数据连接就从NEW变为ESTABLISHED,而且该状态会继续匹配这个连接后继数据包。

RELATED：

当一个连接和某个已处于ESTABLISHED状态的连接有关系时，就被认为是RELATED，也就是说，一个连接想要是RELATED的，首先要有个ESTABLISHED的连接，这个ESTABLISHED连接再产生一个主连接之外的连接，这个新的连接就是RELATED。

INVALID：

INVALID状态说明数据包不能被识别属于哪个连接或没有任何状态。

例：

对本机22端口做状态监测：

进来的请求状态为new，而出去的状态则为ESTABLISHED，如果自动连接别人 状态肯定为NEW，如果正常去响应别人那么状态肯定是ESTABLISHED

[root@test3~]# iptables -I INPUT -s 10.0.10.0/24 -d 10.0.10.62 -p tcp --dport 22 -m state--state NEW,ESTABLISHED -j ACCEPT

出口的响应都必须是ESTABLISHED

[root@test3~]# iptables -A OUTPUT -s 10.0.10.62 -d 10.0.10.0/24 -p tcp --dport 22 -m state--state ESTABLISHED -j ACCEPT

[root@test3~]# iptables -L -n

ChainINPUT (policy ACCEPT)

target prot opt source destination

ACCEPT tcp -- 10.0.10.0/24 10.0.10.62 tcp dpt:22 state NEW,ESTABLISHED

ChainFORWARD (policy DROP)

target prot opt source destination

ChainOUTPUT (policy ACCEPT)

target prot opt source destination

ACCEPT tcp -- 10.0.10.62 10.0.10.0/24 tcp dpt:22state ESTABLISHED

多端口规则匹配

使用参数-m multiport 可以指定15个以内的非连续端口，比如21-22,80

-mmulitport

--src-prots

--dst-ports

--prots

#对多端口进行匹配，只要匹配以下端口，则全部放行

[root@test3~]# iptables -A INPUT -s 10.0.10.0/24 -d10.0.10.62 -p tcp -m state --state NEW -m mulitport--destination-ports 21,22,80 -j ACCEPT

多IP匹配,指定匹配的IP地址范围：

-miprange

--src-range

--dst-range

指定匹配的连续ip段

[root@test3~]# iptables -A INPUT -s -m iprange --src-range 10.0.10.100-10.0.10.200

指定速率匹配

默认为每秒匹配3个报文，基于令牌桶算法

-mlimit

--limit #NUMBER，表示允许收集多少个空闲令牌

--limit-burst #RATE，允许放行多少个报文

比如：ssh一分钟之内只能建立20个链接，平均5秒一个，而一次性只能放行2个空闲令牌

--limit 20/min

--limit-burst 2

只有在大量空闲令牌存储的情况下，才可有limit-burst控制

例：控制NEW状态的请求

[root@test3~]# iptables -A INPUT -s 10.0.10.0/24 -d 10.0.10.62 -m state --state NEW -mlimit --limit 12/min --limit 12/min --limit-burst 2 -j ACCEPT

例2：每次只允许2个ping包进来

[root@test3~]# iptables -F

[root@test3~]# iptables -A INPUT -s 10.0.10.0/24 -d 10.0.10.62 -p icmp --icmp-type 8 -mlimit --limit 20/min --limit-burst 5 -j ACCEPT

新建立一终端，在其终端ping10.0.10.62可以看到效果，不再演示

2.5对应用层进行匹配

对应用层编码字符串做相似匹配，常用算法使用--alog来指定 ，一般来讲算法一般为bm和kmp

-msrting

--string ""

--algo {bm|kmp}

例：

·假如我们期望web站点页面中任何包含"hello"的字符串的页面，则禁止访问，其他则放行

·请求报文中不会包含hello，一般来讲只包含访问某个页面，那么请求内容无非包含了请求某个链接而已

·响应报文中会封装页面的内容信息，因此 会出现在响应报文中，而不是请求报文

启动httpd服务

[root@test3~]# /etc/init.d/httpd start

在web站点新建页面1.html，内容为"hello" ， 2.html内容为"word"

[root@test3domian]# echo hello > 1.html

[root@test3domian]# echo word > 2.html

在iptables的允许放行规则前面加一条更严谨的禁止规则：

[root@test3domian]# iptables -A OUTPUT -s 10.0.10.62 -p tcp --sport 80 -m string --string"hello" --algo kmp -j REJECT

再次访问

[root@test3domian]# curl -dump http://10.0.10.62/2.html

word

[root@test3domian]# curl -dump http://10.0.10.62/1html

#请求已发出去但是一直没有反应，我们来看一下防火墙规则是否被匹配到

[root@test3domian]# iptables -L -nv

ChainINPUT (policy ACCEPT 255 packets, 30024 bytes)

pkts bytes target prot opt in out source destination

ChainFORWARD (policy DROP 0 packets, 0 bytes)

pkts bytes target prot opt in out source destination

ChainOUTPUT (policy ACCEPT 201 packets, 29406 bytes)

pkts bytes target prot opt in out source destination

35 11209 REJECT tcp -- \* \* 10.0.10.62 0.0.0.0/0 tcp spt:80STRING match "hello" ALGO name kmp TO 65535 reject-withicmp-port-unreachable

基于时间限定

-m time

#指定日期起止范围

--datestart

--datestop

#指定时间的起止范围

--timestart

--timestop

#指定星期x范围

--weekdays

#指定月份

--monthdays

3.基于iptables实现NAT功能

3.1基于SNAT功能的实现

考虑场景：为解决IP地址不足，所以用NAT功能来实现成本节约

SNAT：源地址转换（代理内部客户端访问外部网络）在POSTROUTING或OUTPUT链上来做规则限制

参数选项：

-j SNAT --to-source IP

-j MASQUERADE

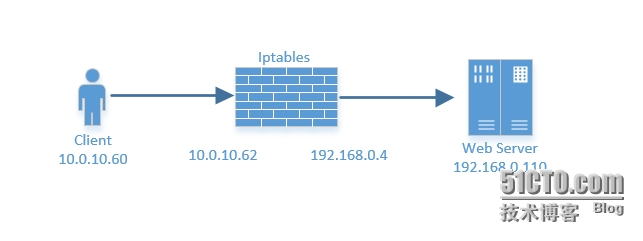
DNAT ：目标地址转换（将内部服务器公开至外部网络）需在PREROUTING做限制

参数选项：

-j DNAT --to-destination IP:prot

NAT不但可以转换目标地址，还可以映射目标端口

拓补图如下：



假设iptables为网关服务器，192.168.0.0为内网地址段 10.0.10.0 为外网地址段

规划：

|  |  |
| --- | --- |
| 服务器角色 | 服务器内网IP地址 |
| iptables | 10.0.10.62、 192.168.0.4 |
| client | 10.0.10.60 |
| web server | 192.168.0.110 |

下面来配置服务器：

webserver服务器配置如下：

[root@mode~]# /etc/init.d/httpd start

[root@modehtml]# echo 111 > test.html

#查看路由信息

[root@modehtml]# route -n

Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface

192.168.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1

10.0.10.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0

169.254.0.0 0.0.0.0 255.255.0.0 U 0 0 0 eth0

0.0.0.0 192.168.0.4 0.0.0.0 UG 0 0 0 eth1

0.0.0.0 192.168.0.1 0.0.0.0 UG 0 0 0 eth1

iptables服务器配置如下：

开启路由转发功能

[root@test3domian]# echo 1 > /proc/sys/net/ipv4/ip\_forward

client配置如下：

#将eth1的网卡关闭，真正意义上断开连接

[root@test~]# ifdown eth1

#添加直连路由

[root@test~]# route add default gw 10.0.10.62

[root@test~]# route -n

KernelIP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface

10.0.10.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0

169.254.0.0 0.0.0.0 255.255.0.0 U 1002 0 0 eth0

0.0.0.0 10.0.10.62 0.0.0.0 UG 0 0 0 eth0

这时去ping192.168.0.0 段的地址是通的，如下所示

[root@test~]# ping 192.168.0.4

PING192.168.0.4 (192.168.0.4) 56(84) bytes of data.

64bytes from 192.168.0.4: icmp\_seq=1 ttl=64 time=22.0 ms

64bytes from 192.168.0.4: icmp\_seq=2 ttl=64 time=0.245 ms

查看是否可访问webserver的web服务

[root@test ~]# curl -dumphttp://192.168.0.110/test.html

111

返回web server上查看访问日志

[root@modelogs]# tail access\_log

10.0.10.60- - [02/Feb/2014:20:33:27 +0800] "POST /test.htmlHTTP/1.1" 200 4 "-" "curl/7.19.7 (x86\_64-redhat-linux-gnu)libcurl/7.19.7 NSS/3.13.6.0 zlib/1.2.3 libidn/1.18 libssh2/1.4.2"

#源地址为10.60 由此可见，由路由来实现两台主机的通信

如果想使用nat方式来实现任何来自10.0/24的网络 通过此台服务器想访问web其他主机，都将源地址改为iptables的ip地址

#凡是来自10.0.10.0网段的主机都将其转换为自己的ip地址

[root@test3domian]# iptables -t nat -A POSTROUTING-s 10.0.10.0/24 -j SNAT --to-source 192.168.0.4

返回client端再次访问web server，并查看日志

[root@mode logs]# tail access\_log

10.0.10.60- - [02/Feb/2014:20:33:27 +0800] "POST /test.html HTTP/1.1" 200 4"-" "curl/7.19.7 (x86\_64-redhat-linux-gnu) libcurl/7.19.7NSS/3.13.6.0 zlib/1.2.3 libidn/1.18 libssh2/1.4.2"

192.168.0.4- - [02/Feb/2014:20:37:13 +0800] "POST /test.htmlHTTP/1.1" 200 4 "-" "curl/7.19.7 (x86\_64-redhat-linux-gnu)libcurl/7.19.7 NSS/3.13.6.0 zlib/1.2.3 libidn/1.18 libssh2/1.4.2"

由此可见，来源IP为代理服务器，因此它是回给192.168.0.4的 ，由192.168.0.4通过nat路由表再返回给client

查看规则

[root@test3domian]# iptables -t nat -L -n -v

ChainPREROUTING (policy ACCEPT 3 packets, 387 bytes)

pkts bytes target prot opt in out source destination

ChainPOSTROUTING (policy ACCEPT 0 packets, 0 bytes)

pkts bytes target prot opt in out source destination

2 144 SNAT all -- \* \* 10.0.10.0/24 0.0.0.0/0 to:192.168.0.4

ChainOUTPUT (policy ACCEPT 0 packets, 0 bytes)

pkts bytes target prot opt in out source destination

将其状态全部放行

[root@test3domian]# iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT

[root@test3domian]# iptables -A FORWARD -s10.0.10.0/24 -p tcp --dport 80 -m state --state NEW -j ACCEPT

再来测试：切换至Client

[root@test~]# curl -dump http://192.168.0.110/test.html

111

返回iptables服务器，查看规则匹配情况

[root@test3domian]# iptables -L -nv

ChainINPUT (policy ACCEPT 45 packets, 3241 bytes)

pkts bytes target prot opt in out source destination

ChainFORWARD (policy ACCEPT 0 packets, 0 bytes)

pkts bytes target prot opt in out source destination

27 2964 ACCEPT all -- \* \* 0.0.0.0/0 0.0.0.0/0 stateESTABLISHED

3 180 ACCEPT tcp -- \* \* 10.0.10.0/24 0.0.0.0/0 tcp dpt:80state NEW

ChainOUTPUT (policy ACCEPT 31 packets, 4064 bytes)

pkts bytes target prot opt in out source destination

通过规则匹配可以看到，首先其会先发起三次握手第一次由第二条规则先匹配到，而后来由状态规则ESTABLISHED进行匹配

3.2定义DNAT的实现

如果构建大并发的环境时，NAT并不适用，一般来讲能够并发用户请求的场景来讲，在2-3W 已经非常庞大了，通常都是专业级硬件分发设备或应用来做分发，下面尝试着使client能够访问web服务器，但期望的是今后访问web服务器不是访问192.168.0.110而是iptables服务器10.0.10.62

因为是实验环境，所以清空所有规则

[root@test3~]# iptables -t nat -F

[root@test3~]# iptables -F

[root@test3~]# iptables -P FORWARD ACCEPT

我们期望网关10.0.10.62为用户访问目标，而不是192.168.0.110，但62上是没有web服务的，所以有人访问62的web服务必须将其转换到110上

所以要在iptables服务器上操作：

[root@test3~]# iptables -t nat -A PREROUTING -d 10.0.10.62 -p tcp --dport 80 -j DNAT--to-destination 192.168.0.110

在客户端测试：

[root@test~]# curl -dump http://10.0.10.62/test.html

111

如果将FORWARD链的规则改为DROP那么该如何来实现：

[root@test3~]# iptables -P FORWARD DROP

再次测试，果然无法访问

它可以实现地址转换，但是地址转换后的报文是无法再转发至内部主机中去，因为forward规则给挡住了

可以将已经建立过连接的请求全部放行，于是乎：

[root@test3~]# iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

[root@test3 ~]# iptables -A FORWARD -d 192.168.0.110 -p tcp --dport 80 -m state--state NEW -j ACCEPT

#目标地址必须是内部web地址 因为forward链在路由之后才可知其是否要进行转发

#也就是意味着用户的请求到达PREROUTING的时候，目标的地址已经改过了，当用户的请求FORWARD之后目标地址已经是web地址

放行转发ssh服务：

我们想请求iptables的22端口则转发至web server端的22端口上去

[root@test3~]# iptables -t nat -A PREROUTING -d 10.0.10.62 -p tcp --dport 22 -j DNAT --to-destination 192.168.0.110

[root@test3~]# iptables -A FORWARD -d 192.168.0.110 -p tcp --dport 22 -m state --state NEW-j ACCEPT

进行登录

[root@test~]# ssh 10.0.10.62

由此可见，以后想登陆10.62则登陆不上去了（可以更改为非22端口等，不再说明了哦）

将80端口请求转发至web端8080端口

更改apache服务的监听端口：

Listen8080

切换至iptables服务端添加规则：

[root@test3~]# iptables -t nat -A PREROUTING -d 10.0.10.62 -p tcp --dport 80 -j DNAT--to-destination 192.168.0.110:8080

[root@test3~]# iptables -t nat -L -nv

Chain PREROUTING (policy ACCEPT 2 packets, 458 bytes)

pkts bytes target prot opt in out source destination

6 360 DNAT tcp -- \* \* 0.0.0.0/0 10.0.10.62 tcp dpt:80to:192.168.0.110:8080

Chain POSTROUTING (policy ACCEPT 9 packets, 564 bytes)

pkts bytes target prot opt in out source destination

Chain OUTPUT (policy ACCEPT 3 packets, 204 bytes)

pkts bytes target prot opt in out source destination

在端口映射环境下如何放行web服务：

在做地址转发的时候必须以转换之后的端口和地址为目标端口和目标ip地址

[root@test3~]# iptables -P FORWARD DROP

[root@test3 ~]# iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT

[root@test3 ~]# iptables -A FORWARD -d 192.168.0.110 -p tcp --dport 8080 -mstate --state NEW -j ACCEPT

假设在此服务器上还有mysql服务，假设mysql在内网工作在正常服务端口，但告知外面工作在6789端口上，那么

进入mysql并授权

mysql>grant all on \*.\* to 'root'@'%' identified by '123456';

mysql>flush privileges;

在iptables服务器上添加规则如下

[root@test3 ~]# iptables -t nat -A PREROUTING-d 10.0.10.62 -p tcp --dport 6789 -j DNAT --to-destination 192.168.0.110:3306

[root@test3~]# iptables -A FORWARD -d 192.168.0.110 -p tcp --dport 3306 -m state --stateNEW -j ACCEPT

[root@test~]# mysql -uroot -h10.0.10.62 -P6789 -p

切换至client端进行测试

[root@test~]# mysql -uroot -h10.0.10.62 -P6789 -p

4.ip\_conntrack 功能

其扩展模块路径为

/proc/net/nf\_conntrack

/proc/sys/net/nf\_conntrack\_max

不同版本的值和相关信息未必一致

[root@test3~]# cat /proc/sys/net/nf\_conntrack\_max

31860

比起红帽5的值已经大的太多了

#这些超时时间非常长,如下所示：

[root@test3 ~]# cat/proc/net/nf\_conntrack

ipv4 2 tcp 6 431999ESTABLISHED src=10.0.10.62 dst=10.0.10.1sport=22 dport=59448 src=10.0.10.1 dst=10.0.10.62 sport=59448 dport=22[ASSURED] mark=0 secmark=0 use=2

#可以在某些时候将值尽量调低，如果不尽量追踪过长时间

[root@test3~]# cat /proc/sys/net/netfilter/nf\_conntrack\_tcp\_timeout\_established

432000 #5天时间

如果没有特殊需求的话，将其值调低，这样可以保证被追踪的连接尽早被记录的文件中清除出去，已腾出空间供其他被追踪使用

既然追踪连接功能有限制而且对性能有很大的影响为何还要开启呢？

启动其功能的原因在于 在某些场景中必须追踪状态，才能达到目的，如果在并发连接非常大的场景下启用连接追踪是不明智的

因此需自己判断好应用场景，不得不启用，连接数也非常大，方法如下：

（1）.调小nf\_conntrack\_tcp\_timeout\_established

（2）.调大 /proc/sys/net/nf\_conntrack\_max #需要一定的内存容量，只要空间充足即可

扩展模块connlimit：

connlimit 连接数限制，一般可以实现控制某源ip地址发起来某连接个数的

--connlimit-above[number] #连接数的上限，如果某个连接数的个数超过为某个值之后（高于），通常用取反的方法来放行：

#iptables-A INPUT -s 10.0.10.0/24 -p tcp --dport 80 -m connlimit ! --connlimit-above 5-j ACCEPT

hashlimit,limit #能够分析每个ip地址的速率

5.recent模块

利用iptables的recent模块来抵御DOS攻击: 22，建立一个列表，保存有所有访问过指定的服务的客户端IP 对本机ssh: 远程连接

(1).利用connlimit模块将单IP的并发设置为3；会误杀使用NAT上网的用户，可以根据实际情况增大该值；

iptables-I INPUT -p tcp --dport 22 -m connlimit --connlimit-above 3 -j DROP

第二句是记录访问tcp 22端口的新连接，记录名称为SSH --set 记录数据包的来源IP，如果IP已经存在将更新已经存在的条目

(2).利用recent和state模块限制单IP在300s内只能与本机建立2个新连接。被限制五分钟后即可恢复访问

iptables -I INPUT -p tcp --dport 22 -m state --state NEW -m recent--set --name SSH

iptables -I INPUT -p tcp --dport 22 -m state --state NEW -m recent--update --seconds 300 --hitcount 3 --name SSH -j LOG --log-prefix "SSHAttach: "

iptables -I INPUT -p tcp --dport 22 -m state --state NEW -m recent--update --seconds 300 --hitcount 3 --name SSH -j DROP

第三句是指SSH记录中的IP，300s内发起超过3次连接则拒绝此IP的连接。

--update 是指每次建立连接都更新列表；

--seconds必须与--rcheck或者--update同时使用

--hitcount必须与--rcheck或者--update同时使用

(3).iptables的记录：/proc/net/xt\_recent/SSH

也可以使用下面的这句记录日志：

iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent--update --name SSH --second 300 --hitcount 3 -j LOG --log-prefix "SSHAttack"