如何自定义iptables规则

1 清除iptables的策略

iptables –F

iptables –Z

iptables –X

2 然后定义默认规则，INPUT一般是DROP，OUTPUT和FORWAR一般是 ACCEPT

iptables –P INPUT DROP

iptables –P OUTPUT ACCEPT

iptables –P FORWARD ACCEPT

3 然后就是定义普通规则，这一般根据自己需要开放不同端口，状态的 packet，但是一定要注意开放lo的设备。

iptables –A INPUT –i lo –j ACCEPT

iptables –A INPUT –p tcp –dport 80 –m state --state NEW,ESTABLISHED, RELATED –j ACCEPT

iptables –A INPUT –p tcp --dport 22 –j ACCEPT

iptables定义规则:

格式：iptables [-t table] COMMAND chain CRETIRIA -j ACTION

-t table ：3个filter nat mangle

COMMAND：定义如何对规则进行管理

chain：指定你接下来的规则到底是在哪个链上操作的，当定义策略的时候，是可以省略的

CRETIRIA:指定匹配标准

-j ACTION :指定如何进行处理

比如：不允许172.16.0.0/24的进行访问。

iptables -t filter -A INPUT -s 172.16.0.0/16 -p udp --dport 53 -j DROP

当然你如果想拒绝的更彻底：

iptables -t filter -R INPUT 1 -s 172.16.0.0/16 -p udp --dport 53 -j REJECT

iptables -L -n -v #查看定义规则的详细信息

**iptables的模块ip\_conntrack，能够追踪每个链接， /proc/net/ip\_conntrack该文件显示当前追踪数目。/proc/sys/net/ipv4/ip\_conntrack\_max该文件设置最大追踪数目，超过后的连接将被丢弃。**

**详解**[**请点击**](../解决问题/ip-conntrack.docx)

**四：详解COMMAND:**

**1.链管理命令（这都是立即生效的）**

-P :设置默认策略的（设定默认门是关着的还是开着的）

默认策略一般只有两种

iptables -P INPUT (DROP|ACCEPT)  默认是关的/默认是开的

比如：

iptables -P INPUT DROP 这就把默认规则给拒绝了。并且没有定义哪个动作，所以关于外界连接的所有规则包括Xshell连接之类的，远程连接都被拒绝了。

        -F: FLASH，清空规则链的(注意每个链的管理权限)

    iptables -t nat -F PREROUTING

    iptables -t nat -F 清空nat表的所有链

        -N:NEW 支持用户新建一个链

            iptables -N inbound\_tcp\_web 表示附在tcp表上用于检查web的。

        -X: 用于删除用户自定义的空链

            使用方法跟-N相同，但是在删除之前必须要将里面的链给清空昂了

        -E：用来Rename chain主要是用来给用户自定义的链重命名

            -E oldname newname

         -Z：清空链，及链中默认规则的计数器的（**有两个计数器，被匹配到多少个数据包，多少个字节**）

            iptables -Z :清空

**2.规则管理命令**

         -A：追加，在当前链的最后新增一个规则

         -I num : 插入，把当前规则插入为第几条。

            -I 3 :插入为第三条

         -R num：Replays替换/修改第几条规则

            格式：iptables -R 3 …………

         -D num：删除，明确指定删除第几条规则

**3.查看管理命令 “-L”**

附加子命令

-n：以数字的方式显示ip，它会将ip直接显示出来，如果不加-n，则会将ip反向解析成主机名。

-v：显示详细信息

-vv

-vvv :越多越详细

-x：在计数器上显示精确值，不做单位换算

--line-numbers : 显示规则的行号

-t nat：显示所有的关卡的信息

**五：详解匹配标准**

**1.通用匹配：源地址目标地址的匹配**

-s：指定作为源地址匹配，这里不能指定主机名称，必须是IP

IP | IP/MASK | 0.0.0.0/0.0.0.0

而且地址可以取反，加一个“!”表示除了哪个IP之外

-d：表示匹配目标地址

-p：用于匹配协议的（这里的协议通常有3种，TCP/UDP/ICMP）

-i eth0：从这块网卡流入的数据

流入一般用在INPUT和PREROUTING上

-o eth0：从这块网卡流出的数据

流出一般在OUTPUT和POSTROUTING上

**2.扩展匹配**

2.1隐含扩展：对协议的扩展

    -p tcp :TCP协议的扩展。一般有三种扩展

--dport XX-XX：指定目标端口,不能指定多个非连续端口,只能指定单个端口，比如

--dport 21  或者 --dport 21-23 (此时表示21,22,23)

--sport：指定源端口

--tcp-fiags：TCP的标志位（SYN,ACK，FIN,PSH，RST,URG）

    对于它，一般要跟两个参数：

1.检查的标志位

2.必须为1的标志位

--tcpflags syn,ack,fin,rst syn   =    --syn

表示检查这4个位，这4个位中syn必须为1，其他的必须为0。所以这个意思就是用于检测三次握手的第一次包的。对于这种专门匹配第一包的SYN为1的包，还有一种简写方式，叫做--syn

    -p udp：UDP协议的扩展

        --dport

        --sport

    -p icmp：icmp数据报文的扩展

        --icmp-type：

echo-request(请求回显)，一般用8 来表示

所以 --icmp-type 8 匹配请求回显数据包

echo-reply （响应的数据包）一般用0来表示

2.2显式扩展（-m）

     扩展各种模块

      -m multiport：表示启用多端口扩展

      之后我们就可以启用比如 --dports 21,23,80

**六：详解-j ACTION**

常用的ACTION：

DROP：悄悄丢弃

一般我们多用DROP来隐藏我们的身份，以及隐藏我们的链表

REJECT：明示拒绝

ACCEPT：接受

custom\_chain：转向一个自定义的链

DNAT

SNAT

MASQUERADE：源地址伪装

REDIRECT：重定向：主要用于实现端口重定向

MARK：打防火墙标记的

RETURN：返回

在自定义链执行完毕后使用返回，来返回原规则链。

**练习题1：**

     只要是来自于172.16.0.0/16网段的都允许访问我本机的172.16.100.1的SSHD服务

     分析：首先肯定是在允许表中定义的。因为不需要做NAT地址转换之类的，然后查看我们SSHD服务，在22号端口上，处理机制是接受，对于这个表，需要有一来一回两个规则，如果我们允许也好，拒绝也好，对于访问本机服务，我们最好是定义在INPUT链上，而OUTPUT再予以定义就好。(会话的初始端先定义)，所以加规则就是：

     定义进来的： iptables -t filter -A INPUT -s 172.16.0.0/16 -d 172.16.100.1 -p tcp --dport 22 -j ACCEPT

     定义出去的： iptables -t filter -A OUTPUT -s 172.16.100.1 -d 172.16.0.0/16 -p tcp --dport 22 -j ACCEPT

     将默认策略改成DROP:

 iptables -P INPUT DROP

 iptables -P OUTPUT DROP

 iptables -P FORWARD DROP

七：状态检测：

是一种显式扩展，用于检测会话之间的连接关系的，有了检测我们可以实现会话间功能的扩展

        什么是状态检测？对于整个TCP协议来讲，它是一个有连接的协议，三次握手中，第一次握手，我们就叫NEW连接，而从第二次握手以后的，ack都为1，这是正常的数据传输，和tcp的第二次第三次握手，叫做已建立的连接（ESTABLISHED）,还有一种状态，比较诡异的，比如：SYN=1 ACK=1 RST=1,对于这种我们无法识别的，我们都称之为INVALID无法识别的。还有第四种，FTP这种古老的拥有的特征，每个端口都是独立的，21号和20号端口都是一去一回，他们之间是有关系的，这种关系我们称之为RELATED。

所以我们的状态一共有四种：

        NEW

        ESTABLISHED

        RELATED

        INVALID

所以我们对于刚才的练习题，可以增加状态检测。比如进来的只允许状态为NEW和ESTABLISHED的进来，出去只允许ESTABLISHED的状态出去，这就可以将比较常见的反弹式木马有很好的控制机制。

对于练习题的扩展：

进来的拒绝出去的允许，进来的只允许ESTABLISHED进来，出去只允许ESTABLISHED出去。默认规则都使用拒绝

iptables -L -n --line-number  ：查看之前的规则位于第几行

    改写INPUT

        iptables -R INPUT 2 -s 172.16.0.0/16 -d 172.16.100.1 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT

        iptables -R OUTPUT 1 -m state --state ESTABLISHED -j ACCEPT

    此时如果想再放行一个80端口如何放行呢？

        iptables -A INPUT -d 172.16.100.1 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT

        iptables -R INPUT 1 -d 172.16.100.1 -p udp --dport 53 -j ACCEPT

练习题2**：**

假如我们允许自己ping别人，但是别人ping自己ping不通如何实现呢？

分析：对于ping这个协议，进来的为8（ping），出去的为0(响应).我们为了达到目的，需要8出去,允许0进来

在出去的端口上：iptables -A OUTPUT -p icmp --icmp-type 8 -j ACCEPT

在进来的端口上：iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT

小扩展：对于127.0.0.1比较特殊，我们需要明确定义它

iptables -A INPUT -s 127.0.0.1 -d 127.0.0.1 -j ACCEPT

iptables -A OUTPUT -s 127.0.0.1 -d 127.0.0.1 -j ACCEPT

马哥练习题：

练习：判断下述规则的意义：

# iptables -N clean\_in

# iptables -A clean\_in -d 255.255.255.255 -p icmp -j DROP

# iptables -A clean\_in -d 172.16.255.255 -p icmp -j DROP

# iptables -A clean\_in -p tcp ! --syn -m state --state NEW -j DROP

# iptables -A clean\_in -p tcp --tcp-flags ALL ALL -j DROP

# iptables -A clean\_in -p tcp --tcp-flags ALL NONE -j DROP

# iptables -A clean\_in -d 172.16.100.7 -j RETURN

# iptables -A INPUT -d 172.16.100.7 -j clean\_in

# iptables -A INPUT -i lo -j ACCEPT

# iptables -A OUTPUT -o lo -j ACCEPT

# iptables -A INPUT -i eth0 -m multiport -p tcp --dports 53,113,135,137,139,445 -j DROP

# iptables -A INPUT -i eth0 -m multiport -p udp --dports 53,113,135,137,139,445 -j DROP

# iptables -A INPUT -i eth0 -p udp --dport 1026 -j DROP

# iptables -A INPUT -i eth0 -m multiport -p tcp --dports 1433,4899 -j DROP

# iptables -A INPUT -p icmp -m limit --limit 10/second -j ACCEPT

利用iptables的recent模块来抵御DOS攻击

ssh: 远程连接，

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

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 DROP

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；

2.利用recent和state模块限制单IP在300s内只能与本机建立3个新连接。被限制五分钟后即可恢复访问。

下面对最后两句做一个说明：

1.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.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/ipt\_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 "SSH Attack"

**八：SNAT和DNAT的实现**

由于我们现在IP地址十分紧俏，已经分配完了，这就导致我们必须要进行地址转换，来节约我们仅剩的一点IP资源。那么通过iptables如何实现NAT的地址转换呢？

**1.SNAT基于原地址的转换**

基于原地址的转换一般用在我们的许多内网用户通过一个外网的口上网的时候，这时我们将我们内网的地址转换为一个外网的IP，我们就可以实现连接其他外网IP的功能。

所以我们在iptables中就要定义到底如何转换：

定义的样式：

比如我们现在要将所有192.168.10.0网段的IP在经过的时候全都转换成172.16.100.1这个假设出来的外网地址：

iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -j SNAT --to-source 172.16.100.1

这样，只要是来自本地网络的试图通过网卡访问网络的，都会被统统转换成172.16.100.1这个IP.

那么，如果172.16.100.1不是固定的怎么办？

我们都知道当我们使用联通或者电信上网的时候，一般它都会在每次你开机的时候随机生成一个外网的IP，意思就是外网地址是动态变换的。这时我们就要将外网地址换成 MASQUERADE(动态伪装):它可以实现自动寻找到外网地址，而自动将其改为正确的外网地址。所以，我们就需要这样设置：

         iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -j MASQUERADE

         这里要注意：地址伪装并不适用于所有的地方。

**2.DNAT目标地址转换**

对于目标地址转换，数据流向是从外向内的，外面的是客户端，里面的是服务器端通过目标地址转换，我们可以让外面的ip通过我们对外的外网ip来访问我们服务器不同的服务器，而我们的服务却放在内网服务器的不同的服务器上。

    如何做目标地址转换呢？：

iptables -t nat -A PREROUTING -d 192.168.10.18 -p tcp --dport 80 -j DNAT --todestination 172.16.100.2

        目标地址转换要做在到达网卡之前进行转换,所以要做在PREROUTING这个位置上

九：控制规则的存放以及开启

注意：你所定义的所有内容，当你重启的时候都会失效，要想我们能够生效，需要使用一个命令将它保存起来

1.service iptables save 命令

它会保存在/etc/sysconfig/iptables这个文件中

    2.iptables-save 命令

iptables-save > /etc/sysconfig/iptables

    3.iptables-restore 命令

开机的时候，它会自动加载/etc/sysconfig/iptabels

如果开机不能加载或者没有加载，而你想让一个自己写的配置文件（假设为iptables.2）手动生效的话：

iptables-restore < /etc/sysconfig/iptables.2

则完成了将iptables中定义的规则手动生效

**十：总结**

         Iptables是一个非常重要的工具，它是每一个防火墙上几乎必备的设置，也是我们在做大型网络的时候，为了很多原因而必须要设置的。学好Iptables,可以让我们对整个网络的结构有一个比较深刻的了解，同时，我们还能够将内核空间中数据的走向以及linux的安全给掌握的非常透彻。我们在学习的时候，尽量能结合着各种各样的项目，实验来完成，这样对你加深iptables的配置，以及各种技巧有非常大的帮助。

附加iptables比较好的文章：

[netfilter/iptables全攻略](http://www.linuxso.com/linuxpeixun/10332.html)