Linux 防火墙 iptables 学习笔记（一）入门要领

要在网上传输的数据会被分成许多小的数据包，我们一旦接通了网络，会有很多数据包进入，离开，或者经过我们的计算机。

首先我们要弄明白，防火墙将怎么对待这些数据包。这些数据包会经过一些相应的规则链，比如要进入你的计算机的数据包会首先进入 INPUT 链，从我们的计算机发出的数据包会经过 OUTPUT 链，如果一台计算机做一个网络的网关（处于内网和外网两个网络连接的两台计算机，这两台计算机之间相互通讯的数据包会经过这台计算机，这台计算机即相当于一个路由器），可能会有很多数据经过这台计算机，那么这些数据包必经 FORWARD 链，FORWARD 链即数据转发链。明白了这些“链”的概念我们才能进一步学习使用 iptables。

现在我们再来分析一下 iptables 规则是如何工作的，假如我们要访问网站 www.yahoo.com，我们要对 www.yahoo.com 发出请求，这些数据包要经过 OUTPUT 链，在请求发出前，Linux 的内核会在 OUTPUT 链中检查有没有相应的规则适合这个数据包，如果没有相应的规则，OUTPUT 链还会有默认的规则，或者允许，或者不允许（事实上，不允许有两种，一种是把请求拒绝，告诉发出请示的程序被拒绝；还有一种是丢弃，让请求发出者傻等，直到超时）。如果得到允许，请求就发出了，而 www.yahoo.com 服务器返回的数据包会经过 INPUT 链，当然，INPUT 链中也会有相应的规则等着它。

下面我们介绍几个 iptable 的命令

iptables -L [-t filter]

这条命令是显示当前有什么已经设置好的防火墙规则，可能的显示结果如下：

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

从这里我们可以看出，iptables 有三个链分别是 INPUT OUTPUT 和 FORWARD.

其中

INPUT 是外部数据要进过我们主机的第一外关卡(当然你前面也可以再加硬件防火墙).

OUTPUT 是你的主机的数据送出时要做的过绿卡

FORWARD 是转发 你在 NAT 时才会用到

要设置iptables 主要是对这三条链进行设置,当然也包括-nat的另外三个链 我们以后再说

你要用 iptables 你就得启到它 启动命令 service iptables restart

iptables 的默认设置为 三条链都是 ACCEPT 如下:

iptables -P INPUT ACCEPT

iptables -P OUTPUT ACCEPT

iptables -P FORWARD ACCEPT

以上信息你可以用 iptables -L 看到

总体来说 iptables 可以有二种设置

1.默认允许,拒绝特别的
2.默认拒绝,允许特别的
二者都有自己有特点,从安全角度看 个人偏向于第二种,就是默认拒绝,允许特别的.但 iptalbes 默认是第一种 默认允许,拒绝特别的
你可以用命令改变默认值来达到我们的要求 命令如下
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
你再用 iptables -L 查看一下就会觉得默认值以改了
先来谈炎几个参数 XZFL
-F 清除规则
-X 清除链
-Z 将链的记数的流量清零
一般来说 再创建访问规则时 都会将原有的规则清零 这是一个比较好的习惯,因为某些规则的存在会影响你建的规则.
基本语法:
iptables [-t filter] [-AI INPUT,OUTPUT,FORWARD] [-io interface]
[-p tcp,udp.icmp,all] [-s ip/nerwork] [--sport ports]
[-d ip/netword] [--dport ports] [-j ACCEPT DROP]
以上是 iptables 的基本语法
A 是添加的意思
I 是播入的意思
io 指的是数据要进入或出去所要经过的端口 如 eth1 eth0 pppoe 等
p 你所要指定的协议
-s 指源地址 可是单个 IP 如 192.168.2.6 也可以是一个网络 192.168.2.0/24 还可以 是一个域名 如 163.com 如果你填写的域名系统会自动解析出他的 IP 并在 iptables 里 显示
--sport 来源端口
-d 同-s 相似 只不过他指的是目标地址 也可以是 IP 域名 和网络
--dport 目标端口
-j 执行参数 ACCEPT DROP
注意:如果以有参数存在 则说明全部接受
1 如我要来自己 l0 接口的数据全部接受,我们可以写成这样:
iptables -A INPUT -i lo -j ACCEPT
2 如果我们想接受 192.168.2.6 这个 IP 地址传来的数据我们可以这样写
iptablse -A INPUT -i eth1 -p tcp -s 192.168.2.6 -j ACCEPT
3 如果我们要拒绝来自己 192.168.2.0/24 这个网的 telnet 连接
iptablse -A INPUT -i eth1 -p udp -s 192.168.2.0/24 --sport 23 -j DROP

Linux 防火墙 iptables 学习笔记（二）参数指令

iptables 指令

语法：

iptables [-t table] command [match] [-j target/jump]

-t 参数用来指定规则表，内建的规则表有三个，分别是：nat、mangle 和 filter，当未指定规则表时，则一律视为是 filter。个规则表的功能如下：

nat 此规则表拥有 Prerouting 和 postrouting 两个规则链，主要功能为进行一对一、一对多、多对多等网址转译工作（SNATDNAT），由于转译工作的特性，需进行目的地网址转译的封包，就不需要进行来源网址转译，反之亦然，因此为了提升改写封包的率，在防火墙运作时，每个封包只会经过这个规则表一次。如果我们把封包过滤的规则定义在这个数据表里，将会造成无法对同一包进行多次比对，因此这个规则表除了作网址转译外，请不要做其它用途。

mangle 此规则表拥有 Prerouting、FORWARD 和 postrouting 三个规则链。
除了进行网址转译工作会改写封包外，在某些特殊应用可能也必须去改写封包（TTL、TOS）或者是设定 MARK（将封包作记号，以进行后续的过滤），这时就必须将这些工作定义在 mangle 规则表中，由于使用率不高，我们不打算在这里讨论 mangle 的用法。

filter 这个规则表是预设规则表，拥有 INPUT、FORWARD 和 OUTPUT 三个规则链，这个规则表顾名思义是用来进行封包过滤的理动作（例如：DROP、LOG、ACCEPT 或 REJECT），我们会将基本规则都建立在此规则表中。

常用命令列表：

命令 -A, --append
范例 iptables -A INPUT ...
说明 新增规则到某个规则链中，该规则将会成为规则链中的最后一条规则。
命令 -D, --delete
范例 iptables -D INPUT --dport 80 -j DROP
iptables -D INPUT 1
说明 从某个规则链中删除一条规则，可以输入完整规则，或直接指定规则编号加以删除。
命令 -R, --replace
范例 iptables -R INPUT 1 -s 192.168.0.1 -j DROP
说明 取代现行规则，规则被取代后并不会改变顺序。
命令 -I, --insert
范例 iptables -I INPUT 1 --dport 80 -j ACCEPT
说明 插入一条规则，原本该位置上的规则将会往后移动一个顺位。
命令 -L, --list
范例 iptables -L INPUT
说明 列出某规则链中的所有规则。
命令 -F, --flush
范例 iptables -F INPUT
说明 删除某规则链中的所有规则。
命令 -Z, --zero
范例 iptables -Z INPUT

说明 将封包计数器归零。封包计数器是用来计算同一封包出现次数，是过滤阻断式攻击不可或缺的工具。
命令 -N, --new-chain
范例 iptables -N allowed
说明 定义新的规则链。
命令 -X, --delete-chain
范例 iptables -X allowed
说明 删除某个规则链。
命令 -P, --policy
范例 iptables -P INPUT DROP
说明 定义过滤政策。 也就是未符合过滤条件之封包，预设的处理方式。
命令 -E, --rename-chain
范例 iptables -E allowed disallowed
说明 修改某自订规则链的名称。
常用封包比对参数：
参数 -p, --protocol
范例 iptables -A INPUT -p tcp
说明 比对通讯协议类型是否相符，可以使用 ！运算子进行反向比对，例如：-p！tcp ，意思是指除 tcp 以外的其它类型，包含 udp、icmp ...等。如果要比对所有类型，则可以使用 all 关键词，例如：-p all。
参数 -s, --src, --source
范例 iptables -A INPUT -s 192.168.1.1
说明 用来比对封包的来源 IP，可以比对单机或网络，比对网络时请用数字来表示屏蔽，例如：-s 192.168.0.0/24，比对 IP 时可以使用 ！运算子进行反向比对，例如：-s！192.168.0.0/24。
参数 -d, --dst, --destination
范例 iptables -A INPUT -d 192.168.1.1
说明 用来比对封包的目的地 IP，设定方式同上。
参数 -i, --in-interface
范例 iptables -A INPUT -i eth0
说明 用来比对封包是从哪片网卡进入，可以使用通配字符 + 来做大范围比对，例如：-i eth+ 表示所有的 ethernet 网卡，也以使用 ！运算子进行反向比对，例如：-i！eth0。
参数 -o, --out-interface
范例 iptables -A FORWARD -o eth0
说明 用来比对封包要从哪片网卡送出，设定方式同上。
参数 --sport, --source-port
范例 iptables -A INPUT -p tcp --sport 22
说明 用来比对封包的来源埠号，可以比对单一埠，或是一个范围，例如：--sport 22:80，表示从 22 到 80 埠之间都算是符合件，如果要比对不连续的多个埠，则必须使用 --multiport 参数，详见后文。比对埠号时，可以使用 ！运算子进行反向比对。
参数 --dport, --destination-port
范例 iptables -A INPUT -p tcp --dport 22
说明 用来比对封包的目的地埠号，设定方式同上。
参数 --tcp-flags

范例 iptables -p tcp --tcp-flags SYN,FIN,ACK SYN
说明 比对 TCP 封包的状态旗号，参数分为两个部分，第一个部分列举出想比对的旗号，第二部分则列举前述旗号中哪些有被设，未被列举的旗号必须是空的。TCP 状态旗号包括：SYN（同步）、ACK（应答）、FIN（结束）、RST（重设）、URG（紧急）
PSH（强迫推送） 等均可使用于参数中，除此之外还可以使用关键词 ALL 和 NONE 进行比对。比对旗号时，可以使用 ！ 运算子行反向比对。
参数 --syn
范例 iptables -p tcp --syn
说明 用来比对是否为要求联机之 TCP 封包，与 iptables -p tcp --tcp-flags SYN,FIN,ACK SYN 的作用完全相同，如果使用 !运算子，可用来比对非要求联机封包。
参数 -m multiport --source-port
范例 iptables -A INPUT -p tcp -m multiport --source-port 22,53,80,110
说明 用来比对不连续的多个来源埠号，一次最多可以比对 15 个埠，可以使用 ！ 运算子进行反向比对。
参数 -m multiport --destination-port
范例 iptables -A INPUT -p tcp -m multiport --destination-port 22,53,80,110
说明 用来比对不连续的多个目的地埠号，设定方式同上。
参数 -m multiport --port
范例 iptables -A INPUT -p tcp -m multiport --port 22,53,80,110
说明 这个参数比较特殊，用来比对来源埠号和目的埠号相同的封包，设定方式同上。注意：在本范例中，如果来源端口号为 80 目的地埠号为 110，这种封包并不算符合条件。
参数 --icmp-type
范例 iptables -A INPUT -p icmp --icmp-type 8
说明 用来比对 ICMP 的类型编号，可以使用代码或数字编号来进行比对。请打 iptables -p icmp --help 来查看有哪些代码可用。
参数 -m limit --limit
范例 iptables -A INPUT -m limit --limit 3/hour
说明 用来比对某段时间内封包的平均流量，上面的例子是用来比对：每小时平均流量是否超过一次 3 个封包。 除了每小时平均次外，也可以每秒钟、每分钟或每天平均一次，默认值为每小时平均一次，参数如后： /second、 /minute、/day。 除了进行封
数量的比对外，设定这个参数也会在条件达成时，暂停封包的比对动作，以避免因骇客使用洪水攻击法，导致服务被阻断。
参数 --limit-burst
范例 iptables -A INPUT -m limit --limit-burst 5
说明 用来比对瞬间大量封包的数量，上面的例子是用来比对一次同时涌入的封包是否超过 5 个（这是默认值），超过此上限的封将被直接丢弃。使用效果同上。
参数 -m mac --mac-source
范例 iptables -A INPUT -m mac --mac-source 00:00:00:00:00:01
说明 用来比对封包来源网络接口的硬件地址，这个参数不能用在 OUTPUT 和 Postrouting 规则炼上，这是因为封包要送出到网后，才能由网卡驱动程序透过 ARP 通讯协议查出目的地的 MAC 地址，所以 iptables 在进行封包比对时，并不知道封包会送到个网络接口去。
参数 --mark
范例 iptables -t mangle -A INPUT -m mark --mark 1
说明 用来比对封包是否被表示某个号码，当封包被比对成功时，我们可以透过 MARK 处理动

作，将该封包标示一个号码，号码最不可以超过 4294967296。
参数 -m owner --uid-owner
范例 iptables -A OUTPUT -m owner --uid-owner 500
说明 用来比对来自本机的封包，是否为某特定使用者所产生的，这样可以避免服务器使用 root 或其它身分将敏感数据传送出，可以降低系统被骇的损失。可惜这个功能无法比对出来自其它主机的封包。
参数 -m owner --gid-owner
范例 iptables -A OUTPUT -m owner --gid-owner 0
说明 用来比对来自本机的封包，是否为某特定使用者群组所产生的，使用时机同上。
参数 -m owner --pid-owner
范例 iptables -A OUTPUT -m owner --pid-owner 78
说明 用来比对来自本机的封包，是否为某特定行程所产生的，使用时机同上。
参数 -m owner --sid-owner
范例 iptables -A OUTPUT -m owner --sid-owner 100
说明 用来比对来自本机的封包，是否为某特定联机（Session ID）的响应封包，使用时机同上。
参数 -m state --state
范例 iptables -A INPUT -m state --state RELATED,ESTABLISHED
说明 用来比对联机状态，联机状态共有四种：INVALID、ESTABLISHED、NEW 和 RELATED。

INVALID 表示该封包的联机编号（Session ID）无法辨识或编号不正确。
ESTABLISHED 表示该封包属于某个已经建立的联机。
NEW 表示该封包想要起始一个联机（重设联机或将联机重导向）。
RELATED 表示该封包是属于某个已经建立的联机，所建立的新联机。例如：FTP-DATA 联机必定是源自某个 FTP 联机。

常用的处理动作：
-j 参数用来指定要进行的处理动作，常用的处理动作包括：ACCEPT、REJECT、DROP、REDIRECT、MASQUERADE、LOG、DNAT、

SNAT、MIRROR、QUEUE、RETURN、MARK，分别说明如下：
ACCEPT 将封包放行，进行完此处理动作后，将不再比对其它规则，直接跳往下一个规则炼（natostrouting）。
REJECT 拦阻该封包，并传送封包通知对方，可以传送的封包有几个选择：ICMP port-unreachable、ICMP echo-reply 或是
tcp-reset（这个封包会要求对方关闭联机），进行完此处理动作后，将不再比对其它规则，直接 中断过滤程序。 范例如下：
iptables -A FORWARD -p TCP --dport 22 -j REJECT --reject-with tcp-reset
DROP 丢弃封包不予处理，进行完此处理动作后，将不再比对其它规则，直接中断过滤程序。
REDIRECT 将封包重新导向到另一个端口（PNAT），进行完此处理动作后，将 会继续比对其它规则。 这个功能可以用来实作通透式
porxy 或用来保护 web 服务器。例如：iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080
MASQUERADE 改写封包来源 IP 为防火墙 NIC IP，可以指定 port 对应的范围，进行完此处理动作后，直接跳往下一个规则（mangleostrouting）。这个功能与 SNAT 略有不同，当

进行 IP 伪装时，不需指定要伪装成哪个 IP，IP 会从网卡直接读，当使用拨接连线时，IP 通常是由 ISP 公司的 DHCP 服务器指派的，这个时候 MASQUERADE 特别有用。范例如下：

iptables -t nat -A POSTROUTING -p TCP -j MASQUERADE --to-ports 1024-31000

LOG 将封包相关讯息纪录在 /var/log 中，详细位置请查阅 /etc/syslog.conf 组态档，进行完此处理动作后，将会继续比对其规则。例如：

iptables -A INPUT -p tcp -j LOG --log-prefix "INPUT packets"

SNAT 改写封包来源 IP 为某特定 IP 或 IP 范围，可以指定 port 对应的范围，进行完此处理动作后，将直接跳往下一个规则（mangleostrouting）。范例如下：

iptables -t nat -A POSTROUTING -p tcp-o eth0 -j SNAT --to-source
194.236.50.155-194.236.50.160:1024-32000

DNAT 改写封包目的地 IP 为某特定 IP 或 IP 范围，可以指定 port 对应的范围，进行完此处理动作后，将会直接跳往下一个规炼（filter:input 或 filter:forward）。范例如下：

iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT
--to-destination
192.168.1.1-192.168.1.10:80-100

MIRROR 镜射封包，也就是将来源 IP 与目的地 IP 对调后，将封包送回，进行完此处理动作后，将会中断过滤程序。

QUEUE 中断过滤程序，将封包放入队列，交给其它程序处理。透过自行开发的处理程序，可以进行其它应用，例如：计算联机费.....等。

RETURN 结束在目前规则炼中的过滤程序，返回主规则炼继续过滤，如果把自订规则炼看成是一个子程序，那么这个动作，就相当提早结束子程序并返回到主程序中。

MARK 将封包标上某个代号，以便提供作为后续过滤的条件判断依据，进行完此处理动作后，将会继续比对其它规则。范例如下：

iptables -t mangle -A PREROUTING -p tcp --dport 22 -j MARK --set-mark 2

Linux 防火墙 iptables 学习笔记（三）iptables 命令详解和举例

网上看到这个配置讲解得还比较易懂，就转过来了，大家一起看下，希望对您工作能有所帮助。

网管员的安全意识要比空喊 Linux 安全重要得多。

```
iptables -F
iptables -X
iptables -F -t mangle
iptables -t mangle -X
iptables -F -t nat
iptables -t nat -X
```

首先，把三个表清空，把自建的规则清空。

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

设定 INPUT、OUTPUT 的默认策略为 DROP，FORWARD 为 ACCEPT。

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

先把"回环"打开，以免有不必要的麻烦。

```
iptables -A INPUT -i eth+ -p icmp --icmp-type 8 -j ACCEPT
iptables -A OUTPUT -o eth+ -p icmp --icmp-type 0 -j ACCEPT
```

在所有网卡上打开 ping 功能，便于维护和检测。

```
iptables -A INPUT -i eth0 -s 192.168.100.250 -d 192.168.100.1 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o eth0 -d 192.168.100.250 -s 192.168.100.1 -p tcp --sport 22 -j ACCEPT
```

打开 22 端口，允许远程管理。（设定了很多的附加条件：管理机器 IP 必须是 250，并且必须从 eth0 网卡进入）

```
iptables -A INPUT -i eth0 -s 192.168.100.0/24 -p tcp --dport 3128 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -d 192.168.100.0/24 -p tcp --sport 3128 -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth1 -s 192.168.168.0/24 -p tcp --dport 3128 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth1 -d 192.168.168.0/24 -p tcp --sport 3128 -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth2 -p tcp --dport 32768:61000 -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth2 -p tcp --sport 32768:61000 -m state --state NEW,ESTABLISHED -j ACCEPT
```

iptables -A OUTPUT -o eth2 -p udp --dport 53 -j ACCEPT
iptables -A INPUT -i eth2 -p udp --sport 53 -j ACCEPT
上面这几句是比较头痛的，我做逐一解释。

iptables -A INPUT -i eth0 -s 192.168.100.0/24 -p tcp --dport 3128 -m state --state NEW,ESTABLISHED -j ACCEPT
允许 192.168.100.0/24 网段的机器发送数据包从 eth0 网卡进入。如果数据包是 tcp 协议，而且目的端口是 3128（因为 REDIRECT 已经把 80 改为 3128 了。nat 表的 PREROUTING 是在 filter 表的 INPUT 前面的。）的，再而且，数据包的状态必须是 NEW 或者 ESTABLISHED 的（NEW 代表 tcp 三段式握手的“第一握”，换句话说就是，允许客户端机器向服务器发出链接申请。ESTABLISHED 表示通过握手已经建立起链接），通过。

iptables -A OUTPUT -o eth2 -p tcp --sport 32768:61000 -m state --state NEW,ESTABLISHED -j ACCEPT
我们先来看这一句。现在你的数据包已经进入到 linux 服务器防火墙上来了。squid 需要代替你去访问，所以这时，服务器就成了客户端的角色，所以它要使用 32768 到 61000 的私有端口进行访问。（大家会奇怪应该是 1024 到 65535 吧。其实 CentOS 版的 linux 所定义的私有端口是 32768 到 61000 的，你可以通过 cat /proc/sys/net/ipv4/ip_local_port_range，查看一下。）再次声明：这里是 squid 以客户端的身份去访问其他的服务器，所以这里的源端口是 32768:61000，而不是 3128！

iptables -A INPUT -i eth2 -p tcp --dport 32768:61000 -m state --state ESTABLISHED -j ACCEPT
当然了，数据有去就有回。

iptables -A OUTPUT -o eth0 -d 192.168.100.0/24 -p tcp --sport 3128 -m state --state ESTABLISHED -j ACCEPT
数据包还得通过服务器，转到内网网卡上。请注意，这里，是 squid 帮你去访问了你想要访问的网站。所以在内网中，你的机器是客户端角色，而 squid 是服务器角色。这与刚才对外访问的过程是不同的。所以在这里，源端口是 3128，而不是 32768:61000。

iptables -A OUTPUT -o eth2 -p udp --dport 53 -j ACCEPT
iptables -A INPUT -i eth2 -p udp --sport 53 -j ACCEPT
当然，DNS 是不可缺少的。

iptables -A INPUT -i eth+ -p tcp --dport 80 -j LOG --log-prefix "iptables_80_alert" --log-level info
iptables -A INPUT -i eth+ -p tcp --dport 21 -j LOG --log-prefix "iptables_21_alert" --log-level info
iptables -A INPUT -i eth+ -p tcp --dport 22 -j LOG --log-prefix "iptables_22_alert" --log-level info
iptables -A INPUT -i eth+ -p tcp --dport 25 -j LOG --log-prefix "iptables_25_alert" --log-level info
iptables -A INPUT -i eth+ -p icmp --icmp-type 8 -j LOG --log-prefix

"iptables_icmp8_alert" --log-level info
当然了，来点日志记录会对网管员有所帮助。

iptables 基本命令使用举例

一、链的基本操作

1、清除所有的规则。

1）清除预设表 filter 中所有规则链中的规则。

iptables -F

2）清除预设表 filter 中使用者自定链中的规则。

#iptables -X

#iptables -Z

2、设置链的默认策略。一般有两种方法。

1）首先允许所有的包，然后再禁止有危险的包通过放火墙。

#iptables -P INPUT ACCEPT

#iptables -P OUTPUT ACCEPT

#iptables -P FORWARD ACCEPT

2）首先禁止所有的包，然后根据需要的服务允许特定的包通过防火墙。

#iptables -P INPUT DROP

#iptables -P OUTPUT DROP

#iptables -P FORWARD DROP

3、列出表/链中的所有规则。默认只列出 filter 表。

#iptables -L

4、向链中添加规则。下面的语句用于开放网络接口：

#iptables -A INPUT -i lo -j ACCEPT

#iptables -A OUTPUT -o lo -j ACCEPT

#iptables -A INPUT -i eth0 -j ACEPT

#iptables -A OUTPUT -o eth1 -j ACCEPT

#iptables -A FORWARD -i eth1 -j ACCEPT

#iptables -A FORWARD -0 eth1 -j ACCEPT

注意:由于本地进程不会经过 FORWARD 链，因此回环接口 lo 只在 INPUT 和 OUTPUT 两个链上作用。

5、使用者自定义链。

#iptables -N custom

#iptables -A custom -s 0/0 -d 0/0 -p icmp -j DROP

#iptables -A INPUT -s 0/0 -d 0/0 -j DROP

二、设置基本的规则匹配

1、指定协议匹配。

1）匹配指定协议。

#iptables -A INPUT -p tcp

2）匹配指定协议之外的所有协议。

#iptables -A INPUT -p !tcp

2、指定地址匹配。

1）指定匹配的主机。

#iptables -A INPUT -s 192.168.0.18

2）指定匹配的网络。

#iptables -A INPUT -s 192.168.2.0/24

3）匹配指定主机之外的地址。

#iptables -A FORWARD -s !192.168.0.19

4）匹配指定网络之外的网络。

#iptables -A FORWARD -s ! 192.168.3.0/24

3、指定网络接口匹配。

1）指定单一的网络接口匹配。

#iptables -A INPUT -i eth0

#iptables -A FORWARD -o eth0

2）指定同类型的网络接口匹配。

#iptables -A FORWARD -o ppp+

4、指定端口匹配。

1）指定单一端口匹配。

#iptables -A INPUT -p tcp --sport www

#iptables -A INPUT -p udp –dport 53

2）匹配指定端口之外的端口。

#iptables -A INPUT -p tcp –dport !22

3）匹配端口范围。

#iptables -A INPUT -p tcp –sport 22:80

4）匹配 ICMP 端口和 ICMP 类型。

#iptables -A INOUT -p icmp –icimp-type 8

5）指定 ip 碎片。

每

个网络接口都有一个 MTU（最大传输单元），这个参数定义了可以通过的数据包的最大尺寸。如果一个数据包大于这个参数值时，系统会将其划分成更小的数据包

（称为 ip 碎片）来传输，而接受方则对这些 ip 碎片再进行重组以还原整个包。这样会导致一个问题：当系统将大数据包划分成 ip 碎片传输时，第一个碎片含有

完整的包头信息（IP+TCP、UDP 和 ICMP），但是后续的碎片只有包头的部分信息（如源地址、目的地址）。因此，检查后面的 ip 碎片的头部（象有

TCP、UDP 和 ICMP 一样）是不可能的。假如有这样的一条规则：

#iptables -A FORWARD -p tcp -s 192.168.1.0/24 -d 192.168.2.100 –dport 80 -j ACCEPT

并且这时的 FORWARD 的 policy 为 DROP 时，系统只会让第一个 ip 碎片通过，而余下的碎片因为包头信息不完整而无法通过。可以通过—fragment/-f 选项来指定第二个及以后的 ip 碎片解决上述问题。

#iptables -A FORWARD -f -s 192.168.1.0/24 -d 192.168.2.100 -j ACCEPT

注意现在有许多进行 ip 碎片攻击的实例，如 DoS 攻击，因此允许 ip 碎片通过是有安全隐患的，对于这一点可以采用 iptables 的匹配扩展来进行限制。

三、设置扩展的规则匹配（举例已忽略目标动作）

1、多端口匹配。

1）匹配多个源端口。

#iptables -A INPUT -p tcp -m multiport –sport 22,53,80,110

2）匹配多个目的端口。

#iptables -A INPUT -p tcp -m multiport –dpoort 22,53,80

3）匹配多端口(无论是源端口还是目的端口）

#iptables -A INPUT -p tcp -m multiport –port 22,53,80,110

2、指定 TCP 匹配扩展

使用 –tcp-flags 选项可以根据 tcp 包的标志位进行过滤。

#iptables -A INPUT -p tcp –tcp-flags SYN,FIN,ACK SYN

#iptables -A FROWARD -p tcp –tcp-flags ALL SYN,ACK

上实例中第一个表示 SYN、ACK、FIN 的标志都检查，但是只有 SYN 匹配。第二个表示 ALL（SYN，ACK，FIN，RST，URG，PSH）的标志都检查，但是只有设置了 SYN 和 ACK 的匹配。

#iptables -A FORWARD -p tcp --syn

选项—syn 相当于”--tcp-flags SYN,RST,ACK SYN”的简写。

3、limit 速率匹配扩展。

1）指定单位时间内允许通过的数据包个数，单位时间可以是/second、/minute、/hour、/day 或使用第一个子母。

#iptables -A INPUT -m limit --limit 300/hour

2)指定触发事件的阀值。

#iptables -A INPUT -m limit –limit-burst 10

用来比对一次同时涌入的封包是否超过 10 个，超过此上限的包将直接丢弃。

3）同时指定速率限制和触发阀值。

#iptables -A INPUT -p icmp -m limit –-limit 3/m –limit-burst 3

表示每分钟允许的最大包数量为限制速率（本例为 3）加上当前的触发阀值 burst 数。任何情况下，都可保证 3 个数据包通过，触发阀值 burst 相当于允许额外的包数量。

4）基于状态的匹配扩展（连接跟踪）

每个网络连接包括以下信息：源地址、目标地址、源端口、目的端口，称为套接字对（socket pairs）；协议类型、连接状态（TCP 协议）

和超时时间等。防火墙把这些信息称为状态（stateful）。状态包过滤防火墙能在内存中维护一个跟踪状态的表，比简单包过滤防火墙具有更大的安全性，命令格式如下：

iptables -m state –-state [!]state [,state,state,state]

其中，state 表是一个逗号分割的列表，用来指定连接状态，4 种：

>NEW：该包想要开始一个新的连接（重新连接或连接重定向）

>RELATED:该包是属于某个已经建立的连接所建立的新连接。举例：

FTP 的数据传输连接和控制连接之间就是 RELATED 关系。

>ESTABLISHED：该包属于某个已经建立的连接。

>INVALID:该包不匹配于任何连接，通常这些包被 DROP。

例如：

（1）在 INPUT 链添加一条规则，匹配已经建立的连接或由已经建立的连接所建立的新连接。即匹配所有的 TCP 回应包。

#iptables -A INPUT -m state –state RELATED,ESTABLISHED

（2）在 INPUT 链链添加一条规则，匹配所有从非 eth0 接口来的连接请求包。

#iptables -A INPUT -m state –-state NEW -i !eth0

又如，对于 ftp 连接可以使用下面的连接跟踪：

（1）被动（Passive）ftp 连接模式。

#iptables -A INPUT -p tcp --sport 1024: --dport 1024: -m state –-state ESTABLISHED -j ACCEPT

#iptables -A OUTPUT -p tcp --sport 1024: --dport 1024: -m

state --state ESTABLISHED,RELATED -j ACCEPT

（2）主动（Active）ftp 连接模式

#iptables -A INNPUT -p tcp --sport 20 -m state --state ESTABLISHED,RELATED -j ACCEPT

#iptables -A OUTPUT -p tcp –OUTPUT -p tcp –dport 20 -m state --state ESTABLISHED -j ACCEPT

Linux 防火墙 iptables 学习笔记（四）iptables 实现 NAT

1.概述

1.1 什么是 NAT

在传统的标准的 TCP/IP 通信过程中，所有的路由器仅仅是充当一个中间人的角色，也就是通常所说的存储转发，路由器并不会对转发的数据包进行修改，更为确切的说，除了将源 MAC 地址换成自己的 MAC 地址以外，路由器不会对转发的数据包做任何修改。NAT(Network Address Translation 网络地址翻译)恰恰是出于某种特殊需要而对数据包的源 ip 地址、目的 ip 地址、源端口、目的端口进行改写的操作。

1.2 为什么要进行 NAT

我们来看看再什么情况下我们需要做 NAT。

假设有一家 ISP 提供园区 Internet 接入服务，为了方便管理，该 ISP 分配给园区用户的 IP 地址都是伪 IP，但是部分用户要求建立自己的 WWW 服务器对外发布信息，这时候我们就可以通过 NAT 来提供这种服务了。我们可以再防火墙的外部网卡上绑定多个合法 IP 地址，然后通过 NAT 技术使发给其中某一个 IP 地址的包转发至内部某一用户的 WWW 服务器上，然后再将该内部 WWW 服务器响应包伪装成该合法 IP 发出的包。

再比如使用拨号上网的网吧，因为只有一个合法的 IP 地址，必须采用某种手段让其他机器也可以上网，通常是采用代理服务器的方式，但是代理服务器，尤其是应用层代理服务器，只能支持有限的协议，如果过了一段时间后又有新的服务出来，则只能等待代理服务器支持该新应用的升级版本。如果采用 NAT 来解决这个问题，

因为只在应用层以下进行处理，不但可以获得很高的访问速度，而且可以无缝的支持任何新的服务或应用。

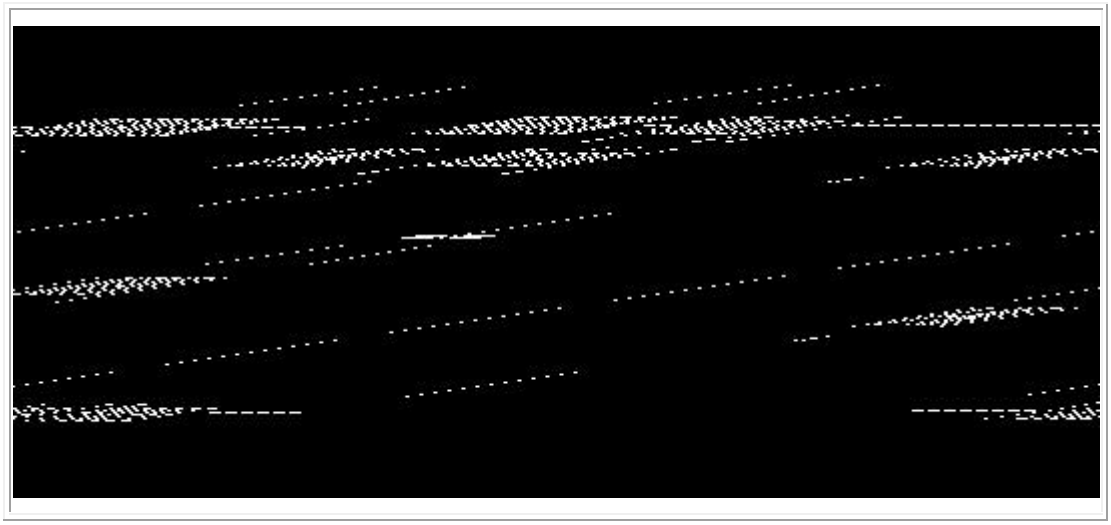
还有一个方面的应用就是重定向，也就是当接收到一个包后，不是转发这个包，而是将其重定向到系统上的某一个应用程序。最常见的应用就是和 squid 配合使用成为透明代理，在对 http 流量进行缓存的同时，可以提供对 Internet 的无缝访问。

1.3 NAT 的类型

在 linux2.4 的 NAT-HOWTO 中，作者从原理的角度将 NAT 分成了两种类型，即源 NAT(SNAT)和目的 NAT(DNAT)，顾名思义，所谓 SNAT 就是改变转发数据包的源地址，所谓 DNAT 就是改变转发数据包的目的地址。

2.原理

下图是 linux2.4 的原理图：



在“用 iptales 实现包过虑型防火墙”一文中我们说过，netfilter 是 Linux 核心中一个通用架构，它提供了一系列的"表"(tables)，每个表由若干"链"(chains)组成，而每条链中可以有一条或数条规则(rule)组成。并且系统缺省的表是"filter"。但是在使用 NAT 的时候，我们所使用的表不再是"filter"，而是"nat"表，所以我们必须使用"-t nat"选项来显式地指明这一点。因为系统缺省的表是"filter"，所以在使用 filter 功能时，我们没有必要显式的指明"-t filter"。

同 filter 表一样，nat 表也有三条缺省的"链"(chains)，这三条链也是规则的容器，它们分别是：

PREROUTING：可以在这里定义进行目的 NAT 的规则，因为路由器进行路由时只检查数据包的目的 ip 地址，所以为了使数据包得以正确路由，我们必须在路由之前就进行目的 NAT;
POSTROUTING：可以在这里定义进行源 NAT 的规则，系统在决定了数据包的路由以后在执行该链中的规则。
OUTPUT：定义对本地产生的数据包的目的 NAT 规则。

3.操作语法

如前所述，在使用 iptables 的 NAT 功能时，我们必须在每一条规则中使用"-t nat"显示的指明使用 nat 表。然后使用以下的选项：

3.1 对规则的操作

加入(append) 一个新规则到一个链 (-A)的最后。
在链内某个位置插入(insert) 一个新规则(-I)，通常是插在最前面。
在链内某个位置替换(replace) 一条规则 (-R)。
在链内某个位置删除(delete) 一条规则 (-D)。
删除(delete) 链内第一条规则 (-D)。

3.2 指定源地址和目的地址

通过--source/--src/-s 来指定源地址(这里的/表示或者的意思，下同)，通过--destination/--dst/-s 来指定目的地址。可以使用以 下四中方法来指定 ip 地址：

使用完整的域名，如“www.linuxaid.com.cn”;

使用 ip 地址，如“192.168.1.1”;
用 x.x.x.x/x.x.x.x 指定一个网络地址，如“192.168.1.0/255.255.255.0”;
用 x.x.x.x/x 指定一个网络地址,如“192.168.1.0/24”这里的 24 表明了子网掩码的有效位数,这是 UNIX 环境中通常使用的表示方法。缺省的子网掩码数是 32，也就是说指定 192.168.1.1 等效于 192.168.1.1/32。

3.3 指定网络接口

可以使用--in-interface/-i 或--out-interface/-o 来指定网络接口。从 NAT 的原理可以看出，对于 PREROUTING 链，我们只能用-i 指定进来的网络接口；而对于 POSTROUTING 和 OUTPUT 我们只能用 -o 指定出去的网络接口。

3.4 指定协议及端口

可以通过--protocol/-p 选项来指定协议，如果是 udp 和 tcp 协议，还可--source-port/--sport 和 --destination-port/--dport 来指明端口。

4.准备工作

4.1 编译内核，编译时选中以下选项，具体可参看“用 iptales 实现包过虑型防火墙”一文:

<M>Full NAT
<M> MASQUERADE target support
<M> REDIRECT target support

4.2 要使用 NAT 表时，必须首先载入相关模块:

modprobe ip_tables
modprobe ip_nat_ftp

iptable_nat 模块会在运行时自动载入。

5.使用实例

5.1 源 NAT(SNAT)

比如，更改所有来自 192.168.1.0/24 的数据包的源 ip 地址为 1.2.3.4:
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j SNAT --to 1.2.3.4

这里需要注意的是，系统在路由及过虑等处理直到数据包要被送出时才进行 SNAT。

有一种 SNAT 的特殊情况是 ip 欺骗，也就是所谓的 Masquerading，通常建议在使用拨号上网的时候使用，或者说在合法 ip 地址不固定的情况下使用。比如
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

可以看出，这时候我们没有必要显式的指定源 ip 地址等信息。

5.2 目的 SNAT(DNAT)

比如，更改所有来自 192.168.1.0/24 的数据包的目的 ip 地址为 1.2.3.4：
iptables -t nat -A PREROUTING -s 192.168.1.0/24 -i eth1 -j DNAT --to 1.2.3.4

这里需要注意的是，系统是先进行 DNAT，然后才进行路由及过虑等操作。

有一种 DNAT 的特殊情况是重定向，也就是所谓的 Redirection，这时候就相当于将符合条件的数据包的目的 ip 地址改为数据包进入系统时的网络接口的 ip 地址。通常是在与 squid 配置形成透明代理时使用，假设 squid 的监听端口是 3128，我们可以通过以下语句来将来自 192.168.1.0/24，目的端口为 80 的数据包重定向到 squid 监听端口：
iptables -t nat -A PREROUTING -i eth1 -p tcp -s 192.168.1.0/24 --dport 80 -j REDIRECT --to-port 3128

6.综合例子

6.1 使用拨号带动局域网上网

小型企业、网吧等多使用拨号网络上网，通常可能使用代理，但是考虑到成本、对协议的支持等因素，建议使用 ip 欺骗方式带动区域网上网。

成功升级内核后安装 iptables，然后执行以下脚本：

```
#载入相关模块
modprobe ip_tables
modprobe ip_nat_ftp
#进行 ip 伪装
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

6.2 ip 映射

假设有一家 ISP 提供园区 Internet 接入服务，为了方便管理，该 ISP 分配给园区用户的 IP 地址都是伪 IP，但是部分用户要求建立自己的 WWW 服务器对外发布信息。我们可以再防火墙的外部网卡上绑定多个合法 IP 地址，然后通过 ip 映射使发给其中某一个 IP 地址的包转发至内部某一用户的 WWW 服务器上，然后再将该内部 WWW 服务器响应包伪装成该合法 IP 发出的包。

我们假设以下情景：

该 ISP 分配给 A 单位 www 服务器的 ip 为：

伪 ip：192.168.1.100

真实 ip：202.110.123.100

该 ISP 分配给 B 单位 www 服务器的 ip 为：

伪 ip：192.168.1.200

真实 ip：202.110.123.200

linux 防火墙的 ip 地址分别为：

内网接口 eth1：192.168.1.1

外网接口 eth0：202.110.123.1

然后我们将分配给 A、B 单位的真实 ip 绑定到防火墙的外网接口，以 root 权限执行以下命令：

```
ifconfig eth0 add 202.110.123.100 netmask 255.255.255.0
ifconfig eth0 add 202.110.123.200 netmask 255.255.255.0
```

成功升级内核后安装 iptables，然后执行以下脚本：

```
#载入相关模块
modprobe ip_tables
modprobe ip_nat_ftp
```

首先，对防火墙接收到的目的 ip 为 202.110.123.100 和 202.110.123.200 的所有数据包进行目的 NAT(DNAT)：

```
iptables -A PREROUTING -i eth0 -d 202.110.123.100 -j DNAT --to 192.168.1.100
iptables -A PREROUTING -i eth0 -d 202.110.123.200 -j DNAT --to 192.168.1.200
```

其次,对防火墙接收到的源 ip 地址为 192.168.1.100 和 192.168.1.200 的数据包进行源 NAT(SNAT)：

```
iptables -A POSTROUTING -o eth0 -s 192.168.1.100 -j SNAT --to 202.110.123.100
iptables -A POSTROUTING -o eth0 -s 192.168.1.200 -j SNAT --to 202.110.123.200
```

这样，所有目的 ip 为 202.110.123.100 和 202.110.123.200 的数据包都将分别被转发给 192.168.1.100 和 192.168.1.200；而所有来自 192.168.1.100 和 192.168.1.200 的数据包都将分别被伪装成由 202.110.123.100 和 202.110.123.200，从而也就实现了 ip 映射。

Linux 防火墙 iptables 学习笔记（五）linux+iptables 构筑防火墙实例

本文旨在用为公司做防火墙的实例，让大家对 Linux+iptables 做防火墙的安装和配置有一个大致的了解，希望能起到抛砖引玉的作用。

系统环境与网络规化

先了解一下公司的环境，公司利用 2M ADSL 专线上网，电信分配公用 IP 为 218.4.62.12/29,网关为 218.4.62.13 ,公司有电脑五十多台，使用 DHCP，IP 是 192.168.2.XXX，DHCP Server 建在 iptables Server 上；另公司有一电脑培训中心，使用指定固定 IP,IP 为 192.168.20.XXX,为了更加快速的浏览网页,我们架了一台 Squid Server，所有电脑通过 Squid Server 浏览网页，公司还另有一台 WEB Server+Mail Server+Ftp Server。其 IP 为 218.4.62.18。以上电脑和服务器要求全架在防火墙内。我们规化如下：

Iptables Server 上有三块网卡,eth0 上加有二个 IP，218.4.62.14 和 218.4.62.18。

其中 218.4.62.14 为共享上网，218.4.62.18 为 WEB Server 专用，Eth1 的 IP 为 192..168.2.9；为了使培训中心 PC 与公司 PC 之间互不访问，所以直接从 Iptables Server 接到 Switch-B,eth2 接至 Switch-A，连接培训中心 PC 和 Squid Server, Web Server。

网络规化好了后，就开始装服务器了，Iptables Server 用的系统为 Redhat Linux V7.3。在装服务器时要注意选上防火墙的安装包。

IPTABLES 基础

Iptables 语法：

Iptables [-t TABLE] ACTION [PATTERN] [-j TARGET]

TABLE:

有 filter,nat,mangle；若无指定，预设为 filter table.

ACTION(对 Chains 执行的动作):

ACTION 说明

-L Chain 显示 Chain 中的所有规则

-A Chain 对 Chain 新增一条规则

-D Chain 删除 Chain 中的一条规则

-I Chain 在 Chain 中插入一条规则

-R Chain 替换 Chain 中的某一条规则

-P Chain 对 Chain 设定的预设的 Policy

-F Chain 清除 Chain 中的所有规则

-N Chain 自订一个 Chain

-X 清除所有的自订 Chain

CHAINS:

Iptables 有五条默认的 Chains(规则链)，如下表:

Chains 发生的时机

PREROUTING 数据包进入本机后，进入 Route Table 前

INPUT 数据包通过 Route Table 后，目地为本机

OUTPUT 由本机发出，进入 Route Table 前

FORWARD 通过 Route Table 后，目地不是本机时

POSTROUTING 通过 Route Table 后，送到网卡前

PATTERN(设定条件部份):

参数 内容 说明

-p Protocol 通讯协议，如 tcp,udp,icmp,all 等。。。

-s Address 指定的 Source Address 为 Address

-d Address 指定的 Destination Address 为 Address

-I Interface 指定数据包进入的网卡

-o Interface 指定数据包输出的网卡

-m Match 指定高级选项，如 mac,state,multiport 等。。。

TARGET(常用的动作):

TARGET 说明

ACCEPT 让这个数据包通过

DROP 丢弃数据包

RETURN 不作对比直接返回

QUEUE 传给 User-Space 的应用软件处理这个数据包

SNAT nat 专用：转译来源地址

DNAT nat 专用：转译目地地址

MASQUERADE nat 专用：转译来源地址成为 NIC 的 MAC

REDIRECT nat 专用：转送到本机的某个 PORT

用/etc/rc.d/init.d/iptables save 可在/etc/sysconfig/中产生一 iptables 文件，大家可以看到，它有三个*号开始的行，其每一个以*号开始的行对应一个 table，以 COMMIT 表示此 table 的结束。可将要定的规则加入到对应的 table 中，如下：

[root@jiaoyuang init.d]# ./iptables saveSaving current rules to /etc/sysconfig/iptables: [OK][root@jiaoyuang init.d]# cat /etc/sysconfig/iptables

Generated by iptables-save v1.2.4 on Sat Sep 28 16:51:22 2002

*mangle

:PREROUTING ACCEPT [61522:8074850]

:OUTPUT ACCEPT [1079:79301]

COMMIT

Completed on Sat Sep 28 16:51:22 2002

Generated by iptables-save v1.2.4 on Sat Sep 28 16:51:22 2002

*nat

:PREROUTING ACCEPT [31850:5091703]

:POSTROUTING ACCEPT [20:1240]

:OUTPUT ACCEPT [12:776]

COMMIT

Completed on Sat Sep 28 16:51:22 2002

Generated by iptables-save v1.2.4 on Sat Sep 28 16:51:22 2002

*filter

:INPUT ACCEPT [61444:8070296]

:FORWARD ACCEPT [34:1984]

:OUTPUT ACCEPT [1079:79301]

COMMIT

安装并启动 IPTABLES

在安装 RedHat Linux V7.3 后，iptables 就已经被安装了，但默认启动的是 ipchains。你在安装时所定义的一些规则也在/etc/sysconfig /ipchains 中被定义。我们需要将其停止，才能启动 iptables(注意：虽然不停止 ipchains 也可以启动 iptables，但这时 iptables 并没有真正的起作用。Ipchains 和 iptables 是两个防火墙，你只能选择一个)。

service ipchains stop (停止 ipchains)

chkconfig --level 2345 ipchains off (使 ipchains 系统启动时不自动启动)

chkconfig --level 2345 iptables on (使 iptables 在系统启动时自动启动)

vi /etc/rc.d/rc.local (编辑 rc.local，将下面四行加到最后)

ifconfig eth0 add 218.4.62.18 netmask 255.255.255.248

modprobe ip_conntrack_ftp

modprobe ip_nat_ftp

echo “1” > /proc/sys/net/ipv4/ip_forward

（第一行是在 eth0 上再加一个 IP：218.4.62.18，因在安装时只能设一个 IP：218.4.62.14。Ip_conntrack_ftp 和 ip_nat_ftp 为 iptables 运得必须的两个模块；最后一行为使开启服务器 IP 转发功能。）

(如果你将 iptables 的模块加到了内核中，以上第二，三行可省略。)

配置 DHCP Server，以便让公司 PC 自动获得 IP 和网关，网关为 192.168.2.9。具体的方法请参见相关资料，本文不作详述。

reboot

重新启动服务器后，Iptables 就已经开始运行了。

配置 IPTABLES

对 iptables 有了一个基本的了解后，我们就可以来配置我们的服务器了。首先要发布我们的 WEB Server,将以下二行加入/etc/sysconfig/iptables 中的 nat table 内：

-A PREROUTING -d 218.4.62.18 -j DNAT --to-destination 192.168.20.254

-A POSTROUTING -s 192.168.2.254 -j SNAT --to-source 218.4.62.18

第一行为将至服务器的所有目地地址为 218.4.62.18 的包都 NAT 为 192.168.2.254，第二行为将至服务器的所有源地址为 192.168.2.254 的包为 NAT 到 218.4.62.18。请把 WEB Server 的网关设为 192.168.20.9。

下面我们将所有从服务器共享出去的包都 SNAT 为 218.4.62.14,就可完成共享上网的功能了：

-A POSTROUTING -s 192.168.0.0/16 -j SNAT --to-source 218.4.62.14

将下面的规则加入到/etc/sysconfig/iptables 中的 filter tables 内：

-A INPUT -p icmp -m icmp --icmp-type 8 -m limit --limit 6/min --limit-burst 2 -j ACCEPT

-A INPUT -p icmp -m icmp --icmp-type 8 -j REJECT --reject-with icmp-port-unreachable

以上两行是为了防止 Dos 攻击做的一个简单的处理，大家对于各种攻击可做出相应的处理。

-A INPUT –i eth0 –m state –state ESTABLISHED,RELATED –j ACCEPT-A INPUT –i eth0 –j DROP

以上两行是做了一个 INPUT 状态防火墙的处理，其主要作用为防止外部的连接和攻击，因其接受 ESTABLISHED,RELATED 状态(一个包分为 NEW，ESTABLISHED,RELATED，INVALID 四种状态)的包，故又不妨碍从本机出去的连接。

由于并不是所有的电脑都可以上网，所以还要对共享上网的电脑做一个限制：

IP 限制：

-A FORWARD –s 192.168.2.0/29 –p udp –m multiport –port 53 –j ACCEPT

-A FORWARD –s 192.168.2.0/29 –p tcp –m multiport –port 3128,110,25 –j ACCEPT

-A FORWARD –s 192.168.20.253 –j ACCEPT

充许 192.168.2.0~192.168.2.7 和 192.168.20.253(squid server)的电脑可上网和发邮件。3128 是 squid server 的 proxy port。我们用它去共享上网，110 为 pop3,25 为 smtp。Udp 的 53 为 DNS 所要的 port。不过由于使用的是 DHCP，可能每次得到的 IP 都不一样，所以我们就要用下面一种 MAC 限制的方法了。

MAC 限制:

-A FORWARD –m mac ––mac XX:XX:XX:XX:XX:XX –p udp –m multiport –port 53 –j ACCEPT

-A FORWARD –m mac ––mac XX:XX:XX:XX:XX:XX –p tcp –m multiport –port 3128,110,25 –j ACCEPT

如上就可通过网卡来控制上网了，但现在电脑高手多多，改一个 MAC 的地址好像也不是什么难事了，怎么办呢？那就用我们的第三种方法吧。

MAC+IP 限制:

更改/etc/dhcpd.conf,如果 MAC 与 IP 绑定:

subnet 192.168.2.0

netmask 255.255.255.0{

range 192.168.2.30 192.168.2.230;

option broadcast-address 192.168.2.255;

option routers 192.168.2.9;

option domain-name-servers 212.132.16.163;

host meeting-room {

hardware ethernet 00:50:ba:c8:4b:3a;

fixed-address 192.168.2.35;

}}

我们的 Iptables 改为：0

-A FORWARD –s 192.168.2.35 –m mac --mac XX:XX:XX:XX:XX:XX –p udp –m multiport –port 53 –j ACCEPT

-A FORWARD –s 192.168.2.35 –m mac --mac XX:XX:XX:XX:XX:XX –p tcp –m multiport –port 3128,110,25 –j ACCEPT

这样做之后，高手也无能为力了，不过公司有位 MM 是兄台的 GF，上班的时候想和她聊聊天，培养培养感情；怎么办呢？我们知道 QQ 用的是 udp 的 4000 端口，如占用则 4002,4003。。。那么就如下了：

-A FORWARD –s 192.168.2.35 –m mac --mac XX:XX:XX:XX:XX:XX –p udp –m multiport –port 53,4000,4001,4002,4003,4004,4005 –j ACCEPT

-A FORWARD –s 192.168.2.35 –m mac --mac XX:XX:XX:XX:XX:XX –p tcp –m multiport –port 3128,110,25 –j ACCEPT

最后加一句：

-A FORWARD –s 192.168.0.0/16 –j DROP

由于前面应该开的都开了，所以最后全部禁止。呵呵，到此大功告成。

总结

世界上没有绝对安全的防火墙，安全永远是相对的。配置 iptables 的思路是先 ACCEPT 再 DROP。共享上网的办法还有一个就是用 iptables server 的 Owner,但由于 linux 没有像 win2k 那样的验证模式，在验证 owner 时有些困难。本人正在测试，但目前还没有比较好的解决办法，哪位兄弟搞定的话请 Mail 小弟，小弟将不胜感激。值得注意的是在做 NAT 时，客户端的网关一定要是 iptagles 的 IP。