Linux安全应用—构建以防火墙为核心的安全管理系统
——NETFILTER/IPTABLES手册
[文档编号 SS-ver001]
上海
电话：13817668606
联系方式：little_eyes@citiz.net
互联网址：暂无

前言
概述
这是一篇以介绍在Linux操作系统平台上构建防火墙系统(Netfilter/Iptables)为主的科技文档，旨在帮助使用者在较短的时间内掌握管理和配置要领，为企业的网络安全提供相关的安全保障。
本文是《Linux安全应用——构建以防火墙为核心的安全管理系统》一文的姐妹篇，如果把那篇文章看成是What is it？那么，本文则以技术细节为主，即How to do？关于为什么要使用基于Linux操作系统平台的防火墙系统的原因，
本文共分为两部分，第一部分具体介绍了Netfilter/Iptables的运行机制和配置管理方法，这是全文中最核心的一部分。第二部分给出了一些具体的范例脚本供系统管理员参考。
鉴于笔者水平有限，文中有错误的地方望不吝赐教。
适用对象
本文首先是为各个企业的网络系统管理员撰写的，无论是那些已经使用了基于Netfilter/Iptables防火墙系统的企业，还是那些正准备使用它的企业，本文的内容都非常适合为系统管理员们提供参考。
对于那些Linux的个人用户和爱好者，本文也不失为一篇相当有价值的参考文档，其中，相当一部分内容深入浅出的讲解能帮助读者很快的理解和掌握Netfilter/Iptables的精髓。
对于希望通过本文了解Netfilter/Iptables的读者，应至少具备一定的Linux操作系统的应用基础，比如文件操作、网络配置操作等，当然，如果使用者还具备一定的编译核心的能力那是在好不过的了。
为了更好的配置防火墙系统，使用者除了掌握Netfilter/Iptables本身的配置管理技巧外，掌握一定的TCP/IP网络知识也是必须的，比如在缺省情况下，应该知道SMTP（Simple Mail Transfer Protocol）协议使用TCP25 端口做为其对外提供服务的端口，FTP（File Transfer Protocol）协议在建立连接的整个过程中，会与客户端建立两条连接，一条是用户传输数据的，另一条则是用户控制传输的。
资源列表
Netfilter/Iptables的官方网站：
http://www.netfilter.org

在Netfilter 的官方网站，用户能够下载Iptables的最新的源代码，Iptables的使用说明文档，FAQ和Mail List，这个站点是有关Netfilter/Iptables最权威、最全面的地方，在这里，你几乎能够找到和Netfilter/iptables相关的所有帮助和技术支持。
术语
文中包含了一些术语，也许读者对其中的一部分似曾相识，但又并不完全理解其正确的含义，或者和其他相关安全产品的概念混淆了。这里有一些解释，并说明了本文中如何使用它们。
DNAT - Destination Network Address Translation 目的网络地址转换。 DNAT是一种改变数据包

目的IP地址的技术，这种技术经常用于将内部网络（RFC1918 定义的地址段）的服务器通过公有的可路由IP地址发布到 Internet上，通过对同一个IP地址分配不同的端口，来决定数据的流向。

SNAT - Source Network Address Translation源网络地址转换。这是一种改变数据包源IP地址的技术，经常用来使多台计算机分享一个Internet地址。这只在IPv4 中使用，因为IPv4 的地址已快用完了，IPv6 将解决这个问题。IPv6 使得地球上每一粒沙子大小的空间都能够分配到一个IP地址，因此IPv6 不存在地址空间短缺的问题。

State - 状态 指明数据包处于什么状态。状态在RFC 793 - Transmission Control Protocol中定义，或由用户在Netfilter/iptables中自定义。需要注意的是Netfilter设定了一些关于连接和数据包的状态，但没有完全使用使用RFC 793 的定义。

User space - 用户空间，指在内核外部或发生在内核外部的任何东西。例如，调用 iptables -h 发生在内核外部，但iptables -A FORWARD -p tcp -j ACCEPT（部分地）发生在内核内部，因为一条新的规则加入了规则集。

Kernel space - 内核空间 ，与用户空间相对，指那些发生在内核内部。

target - 这个词在后文中有大量的应用，它表示对匹配的数据包所做的操作，如ACCEPT、DROP、REDIRECT等等。

约定

本文中涉及的命令、范例都是以Redhat Linux操作系统平台为基础的，尽管在绝大多数的情况下，Linux的大多数命令都是独立于发行版本的，但每个发行版本之间或多或少的存在微小的区别。

本文中的命令以黑体 5 号宋体字表示，如下例：

＃iptables-restore /etc/sysconfig/iptables

本文中涉及到的命令、范例脚本都经过我们的严格测试，能够保证使用者按照文中介绍的做法正确配置防火墙，当然，每一个用户的网络环境都是不一样的，具体的配置还需要用户根据自己的具体情况来适当调整。

致谢

在撰写本文时，参考了Oskar Andreasson的《iptables tutorial 1.1.19》一文，由于本人英语水平有限，因此额外参考了中国Linux公社里的Linux新鲜社员sllscn的翻译稿，在此向他们两人表示谢意。

第一章 构建Netfilter/Iptsbles防火墙系统

## 1.1 获取iptables

iptables可以从http://www.netfilter.org下载，该网站是netfilter/iptables最专业的网站，由Harald Welte、Jozsef Kadlecsik、Martin Josefsson、Patrick McHardy等核心成员进行维护，此外，还有众多的iptables的使用者和开发者为网站提供了大量的文档和更新代码。

## 1.2 编译内核

在一般情况下，我们采用的Linux的distribution比如Redhat都会帮我们安装好Iptables，而且netfilter的核心层也被以模块（Modules）的方式编译进了核心，所以，在绝大多数的情况下，我们是不需要对核心进行重新编译的。

当然，我们写这篇文档的用意决不仅限于教会用户使用几个命令来管理iptables，所以，我们还是来描述一下如何编译核心以使Linux在核心层能够支持数据包过滤。

编译核心的准备工作当然是必需有Linux Kernel的源代码，在源代码文件的目录/usr/src/linux-2.4（这个目录一般是个链接文件）中输入：

＃make menuconfig

会出现如下的画面：

在图中高亮显示的地方回车进入Networking options的核心配置页面，

将高亮显示的部分编译进核心，在该配置页面内，还有一处需要注意，就是Netfilter Configuration的配置页面，在该处回车即可进入配置界面：

从图中可以看到，左面尖括号内的“M”代表该选项被编译成核心模块，仅在系统需要时才被装载入核心空间，由于模块化的核心是不占用核心本身的空间的，因此，对于这些选项，除非肯定不会用到的以外，其他的都可以选择编译成核心模块，但有些是必须的，如

Connection tracking——用于支持状态链接跟踪功能

FTP protocol support——用于支持对Ftp协议的连接跟踪机制

IP tables support——用于支持包过滤

Connection state match support——用户支持连接状态匹配

MASQUERADE target support——支持地址伪装功能

Multiport match support——支持多端口匹配，这对于设置过滤规则非常有好处

REDIRECT target support——如果你想使用iptables将特定的流量交给特定的代理程序如squid来处理，这个选项需要被编译进核心模块

这些选项选择完成后，按照核心编译的方法对配置好的核心进行编译即可：

＃make dep

＃make bzImage

＃make install

＃make modules

＃make modules_install

### 1.3 iptables的编译和安装

就象我们在上节中说到的一样，如果采用诸如Redhat之类的Linux Distribution，在安装系统时，Iptables已经被做为缺省的软件安装到系统上了，用户不需要另行安装，如果用户需要另外安装新版本的iptables软件，可以到http://www.netfilter.org的网站下...本是 1.2.9。

从网站上下载的源代码是“tar.bz2”格式的，编译源代码的第一步工作是解压缩，命令如下：

＃/bin/tar xjvf iptables-1.2.9.tar.bz2

解压缩后，进入源代码的安装目录，开始编译：

＃make KERNEL_DIR=/usr/src/linux/

如果一切正常，那么iptables应该编译好了，接下来可以进行安装了，安装命令非常简单：

＃make install KERNEL_DIR=/usr/src/linux/

怎么样？简单吧，当然，如果这时您的系统核心还没有将netfilter/iptables编译进去，那么安装了iptables软件是没有多大意义的。

1.4 iptables的启动和关闭

在Redhat Linux上，由于历史的原因，ipchains和iptables是并存的（ipchains是在kerenl版本 2.4.x 以前的包过滤防火墙系统），因此ipchains和iptables同时运行是不允许的，我们首先要将ipchains的服务停掉：

#/sbin/chkconfig –level 123456 ipchains off

＃/sbin/service ipchains stop

当然，既然不使用ipchains了，我们也可以将ipchains从系统中移除，使用命令：

＃rpm –e ipchains

第二步是启动Iptables的服务

#/sbin/chkconfig –level 345 iptables on

#/sbin/service iptables start

chkconfig命令表示在系统启动时，ipchains或iptables在相应启动级别的缺省设置，如果是off，则代表

系统启动时不启动ipchains或iptables服务。反之，则启动。

1.5 iptables的工作原理和基础架构

iptables被分为两部分，一部分被称为核心空间，另一部分称为用户空间，在核心空间，iptables从底层实现了数据包过滤的各种功能，比如NAT、状态检测以及高级的数据包的匹配策略等，在用户空间，iptables为用户提供了控制核心空间工作状态的命令集。无论如何，一个数据包都会经过下图所示的路径，并在其中的任何一条路径中被处理。

首先，当一个包进来的时候，也就是从以太网卡进入防火墙，内核首先根据路由表决定包的目标。如果目标主机就是本机，则如下图直接进入INPUT链，再由本地正在等待该包的进程接收，否则，如果从以太网卡进来的包目标不是本机，再看是否内核允许转发包(可用 echo 1 > /proc/sys/net/ipv4/ip_forward 打开转发功能如果不允许转发，则包被DROP掉，如果允许转发，则送出本机，这当中决不经过INPUT或者OUTPUT链，因为路由后的目标不是本机，只被转发规则应用，最后，该linux防火墙主机本身能够产生包，这种包只经过OUTPUT链被送出防火墙。

现在，我们来讨论**为什么iptables叫iptables**，这句话挺别扭是吗？但iptables的名字起的确实如其名，我们可以叫它ip表，**在iptables中共有三类表，分别是mangle、nat和filter。**

**mangle表**从目前来看，他的作用对于满足常规的防火墙应用作用不大，我们在这里不进行具体的描述。

**nat表**的作用在于对数据包的源或目的IP地址进行转换，这种应用也许只会在IPv4 的网络中适用，nat表又可主要分为三条链，如下：

DNAT：DNAT操作主要用在这样一种情况下，你有一个合法的IP地址，要把对防火墙的访问重定向到其他的机子上，比如DMZ。也就是说，我们改变的是目的地址，以使包能重路由到某台主机上。

SNAT： SNAT改变包的源地址，这在极大程度上可以隐藏你的本地网络或者DMZ等。一个很好的例子是我们知道防火墙的外部地址，但必须用这个地址替换本地网络地址。有了这个操作，防火墙就能自动地对包做SNAT，以使LAN能连接到Internet。如果使用类似 192.168.0.0/24 这样的地址，是不会从Internet得到任何回应的。因为RFC1918 定义了这些网络为私有的，只能用于LAN内部。

MASQUERADE： MASQUERADE的作用和SNAT完全一样，只是计算机的负荷稍微多一点。因为对每个匹配的包，MASQUERADE都要查找可用的IP地址，而不象SNAT用的IP地址是配置好的。当然，这也有好处，就是如果我们使用诸如PPPOE等拨号的方式连接Internet，这些地址都是由ISP的随机分配的，这时使用MASQUERADE是非常好的一个解决方案。

**filter 表**用来过滤数据包，我们可以在任何时候匹配包并过滤它们。我们就是在这里根据包的内容对包做DROP或ACCEPT的。当然，我们也可以预先在其他地方做些过滤，但是这个表才是设计用来过滤的。几乎所有的target都可以在这儿使用。

1.6 状态机制

状态机制是iptables中较为特殊的一部分，这也是iptables和比较老的ipchains的一个比较大的区别之一，运行状态机制（连接跟踪）的防火墙称作带有状态机制的防火墙，以下简称为状态防火墙。状态防火墙比非状态防火墙要安全，因为它允许我们编写更严密的规则。

在iptables上一共有四种状态，分别被称为**NEW、ESTABLISHED、INVALID、RELATED**，这四种状态对于TCP、UDP、ICMP三种协议均有效。下面，我们来分别阐述四种状态的特性。

NEW：NEW说明这个包是我们看到的第一个包。意思就是，这是conntrack模块看到的某个连接的第一个包，它即将被匹配了。比如，我们看到一个SYN 包，是我们所留意的连接的第一个包，就要匹配它。

ESTABLISHED： ESTABLISHED已经注意到两个方向上的数据传输，而且会继续匹配这个连接的包。处于ESTABLISHED状态的连接是非常容易理解的。只要发送并接到应答，连接就是ESTABLISHED的了。一个连接要从NEW变为ESTABLISHED，只需要接到应答包即可，不管这个包是发往防火墙的，还是要由防火墙转发的。ICMP的错误和重定向等信息包也被看作是ESTABLISHED，只要它们是我们所发出的信息的应答。

RELATED： RELATED是个比较麻烦的状态。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的了。换句话说，一个连接要想是RELATED的，首先要有一个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的了，当然前提是conntrack模块要能理解RELATED。ftp是个很好的例子，FTP-data 连接就是和FTP-control有关联的，如果没有在iptables的策略中配置RELATED状态，FTP-data的连接是无法正确建立的，还有其他的例子，比如，通过IRC的DCC连接。有了这个状态，ICMP应答、FTP传输、DCC等才能穿过防火墙正常工作。注意，大部分还有一些UDP协议都依赖这个机制。这些协议是很复杂的，它们把连接信息放在数据包里，并且要求这些信息能被正确理解。

INVALID：INVALID说明数据包不能被识别属于哪个连接或没有任何状态。有几个原因可以产生这种情况，比如，内存溢出，收到不知属于哪个连接的ICMP错误信息。一般地，我们DROP这个状态的任何东西，因为防火墙认为这是不安全的东西。

每个状态相对于不同的第四层协议来讲，稍微有些区别，对于TCP协议来说，当防火墙收到第一个数据包，也就是SYN报文时，将该会话标记为NEW状态，在系统的/proc/net/目录下，可以查阅文件ip_conntrack，这是在内存空间里存放防火墙当前状态表的临时文件，对于NEW状态的记录如下：

tcp 6 117 SYN_SENT src=192.168.1.5 dst=192.168.1.35 sport=1031 dport=23 [UNREPLIED] src=192.168.1.35 dst=192.168.1.5 sport=23 dport=1031 use=1

从上面的记录可以看出，SYN_SENT状态被设置了，这说明连接已经发出一个SYN包，但应答还没发送过来，这可从[UNREPLIED]标志看出，当服务器端回应了SYN/ACK包后，状态表改写为：

tcp 6 57 SYN_RECV src=192.168.1.5 dst=192.168.1.35 sport=1031 dport=23 src=192.168.1.35 dst=192.168.1.5 sport=23 dport=1031 use=1

现在我们已经收到了相应的SYN/ACK包，状态也变为SYN_RECV，这说明最初发出的SYN包已正确传输，并且SYN/ACK包也到达了防火墙。 这就意味着在连接的两方都有数据传输，因此可以认为两个方向都有相应的回应。

接下来，TCP三次握手的随后一个报文ACK包也到达了防火墙，防火墙上的状态表变成了：

tcp 6 431999 ESTABLISHED src=192.168.1.5 dst=192.168.1.35 sport=1031 dport=23 src=192.168.1.35 dst=192.168.1.5 sport=23 dport=1031 use=1

现在，我们来看看UDP协议的状态描述方法，从协议本身的特性来看，UDP连接是无状态的，因为它没有任何的连接建立和关闭过程。以某个顺序收到的两个数据包是无法确定它们的发出顺序的。但内核仍然可以对UDP连接设置状态。我们来看看是如何跟踪UDP连接的，以及在核心目录 /proc/net/ip_conntrack的相关记录。

当第一个UDP的数据包到达防火墙后，防火墙在他的状态表中留下了这样的记录：

udp 17 20 src=192.168.1.2 dst=192.168.1.5 sport=137 dport=1025 [UNREPLIED] src=192.168.1.5 dst=192.168.1.2 sport=1025 dport=137 use=1

UNREPLIED代表这是一个状态为NEW的数据包，当这条连接的回应数据包到达防火墙后，防火墙立即将修改这条状态记录：

udp 17 160 src=192.168.1.2 dst=192.168.1.5 sport=137 dport=1025 src=192.168.1.5 dst=192.168.1.2 sport=1025 dport=137 use=1

在这条新的状态记录中，UNREPLIED被删除了，这代表现在防火墙已经建立了一条UDP协议的会话，但这里并没有象TCP协议那样显示 ESTABLISHED标记，这是TCP的状态记录和UDP的状态记录稍微不同的一个地方，当然，还有一个地方需要注意，在测试中，还需要有一些数据包经过，防火墙才会将状态记录改写成：

udp 17 179 src=192.168.1.2 dst=192.168.1.5 sport=137 dport=1025 src=192.168.1.5 dst=192.168.1.2 sport=1025 dport=137 [ASSURED] use=1

ASSURED状态表示当前有数据在进行传输，表面当前连接的状态是ACTIVE的。如果，在这个状态下数据

停止了传输，则这条记录会有一个计时器，也就是记录中的第三个字段，上面这条记录的第三个字段是179，代表当前的ASSURED状态还能够保持179秒，如果还有新的数据包经过，那么计时器会被重新设置成缺省的180秒，如果在180秒内都没有流量，那么这条状态记录就会从状态表中被删除。

最后，我们在来看看Linux kernel是如何标示ICMP协议的状态的，ICMP也是一种无状态协议，它只是用来控制而不是建立连接。ICMP包有很多类型，但只有四种类型有应答包，它们是回显请求和应答（Echo request and reply），时间戳请求和应答（Timestamp request and reply），信息请求和应答（Information request and reply），还有地址掩码请求和应答（Address mask request and reply），这些包有两种状态，NEW和ESTABLISHED 。时间戳请求和信息请求已经废除不用了，回显请求还是常用的，比如ping命令就用的到，地址掩码请求不太常用，但是可能有时很有用并且值得使用。看看下面的图，就可以大致了解ICMP连接的NEW和ESTABLISHED状态了。

如图所示，主机向目标发送一个回显请求，防火墙就认为这个包处于NEW状态。目标回应一个回显应答，防火墙就认为包处于ESTABLISHED了。当回显请求被发送时，ip_conntrack里就有这样的记录了：

icmp 1 25 src=192.168.1.6 dst=192.168.1.10 type=8 code=0 id=33029 [UNREPLIED] src=192.168.1.10 dst=192.168.1.6 type=0 code=0 id=33029 use=1

可以看到，ICMP的记录和TCP、UDP的有点区别，协议名称、超时时间和源、目地址都一样，不同之处在于没有了端口，而新增了三个新的字段：type，code和id。字段type说明ICMP的类型。code说明ICMP的代码，这些代码在附录ICMP类型里有说明。id是 ICMP包的ID。每个ICMP包被发送时都被分配一个ID，接受方把同样的ID 分配给应答包，这样发送方能认出是哪个请求的应答。

[UNREPLIED] 的含义和前面一样，说明数的传输只发生在一个方向上，也就是说未收到应答。再往后，是应答包的源、目地址，还有相应的三个新字段，要注意的是type和 code是随着应答包的不同而变化的，id和请求包的一样。和前面一样，应答包被认为是ESTABLISHED的。然而，在应答包之后，这个ICMP连接就不再有数据传输了。所以，一旦应答包穿过防火墙，ICMP的连接跟踪记录就被销毁了。因此，要想在/proc/ip_conntrack文件中抓到 ICMP协议的状态记录实在不是一件容易的事。您可以用如下的命令来尝试获取这些记录：

#cat /proc/net/ip_conntrack | grep icmp

如果没有输出，那么就不停的重复这个命令，直到发现icmp的记录为止。

以上各种情况，请求被认为NEW，应答是ESTABLISHED。换句话说，就是当防火墙看到一个请求包时，就认为连接处于NEW状态，当有应答时，就是ESTABLISHED状态。

1.7 规则的保存和恢复

iptables提供了两个命令来对策略进行保存和恢复：**iptables-save**和**iptables-restore**，iptables-save用来保存当前内存空间的策略，iptables-restore用来将iptables配置文件的策略写入内存空间。

iptables-save的命令格式非常简单：

#iptables-save –c > /etc/sysconfig/iptables

上面的命令表示将内存中的策略写入/etc/sysconfig/iptables文件中，同时将当前内存中针对每条策略的流量统计值也写入该文件。

#iptables-save –t nat > /etc/sysconfig/iptables

这条命令表明只保存当前内存中的nat表。

#iptables-restore /etc/sysconfig/iptables

这条命令表明将/etc/sysconfig/iptables配置文件中的内容写入内存空间，并覆盖当前内存空间中的所有配置。如果不希望更改当前内存空间中的配置，可以添加-n参数，如下：

#iptables-restore –n /etc/sysconfig/iptables

这表明配置文件只将内存空间中没有的策略添加到内存空间。

另一种运行iptables的方法是使用脚本，使用脚本能够实现一些更灵活的、结构化的配置策略，对于一些

习惯使用脚本管理防火墙的系统管理员来说，这是一个更好的选择，后面，我们会提供一些脚本范例。

1.8 编写详细的规则表

本节我们开始揭开iptables的神秘面纱，当我们真正进入iptables的世界，我们发现，这里的世界原来也很精彩。

iptables的所有命令都是以iptables开头，其总体的命令结构如下：

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

1.8.1 table

**-t table表示当前的策略属于哪个table**，前面，我们提到了，一共有三种table：mangle、filter和nat，由于iptables的主要工作是过滤进出本地网络适配器的数据包，因此，很自然的，**如果一条策略是关于过滤的，那么在缺省情况下，“-t filter”是可以省略的，而mangle和nat是一定要注明的**，在实际的应用环境当中，mangle几乎是用不到的，因此，在本文中，我们着重讨论 nat表和filter表。

1.8.2 command

**command指定iptables 对我们提交的规则要做什么样的操作**。这些操作可能是在某个表里增加或删除一些东西，或做点儿其他什么。以下是iptables可用的command：

命令 -A，--append

范例 iptables –A INPUT……

注解 添加规则

命令 -D，--delete

范例 iptables –D INPUT 8，iptables –D FORWARD –p tcp –s 192.168.1.12 –j ACCEPT

注解 从所选的链中删除规则，有两种方法：一种是以编号来表示被删除的规则，另一种是以整条的规则来匹配策略。

命令 -R，--replace

范例 iptables –R FORWARD 2 –p tcp –s 192.168.1.0 –j ACCEPT

注解 替换相应位置的策略，这时有一点需要注意，如果源或目的地址是以名字而不是以IP地址表示的，如果解析出的IP地址多于一个，那么这条命令是失效的。

命令 -I，--insert

范例 iptables –I FORWARD 2 –p tcp –s 192.168.1.0 –j ACCEPT

注解 这个命令和上面一个命令只差一个参数，而不同之处在于这个命令是在相应的位置前面插入一条命令，而不是替换。

命令 -L，--list

范例 iptables –t nat –L，iptables –L INPUT

注解 列出当前内存空间的策略。

命令 -F，flush

范例 iptables –F，iptables –t nat -F

注解 清空所选的链的配置规则。

命令 -N，--new-chain

范例 iptables –N tcp_allowed

注解 添加新的链，在默认情况下，iptables有ACCEPT、DROP、REJECT、LOG、REDIRECT等，如果希望对数据包做定制的处理，可以自己定义新的链。

命令 -X，--delete-chain

范例 iptables –X tcp_allowed

注解 这条命令用于删除自定义的链。

命令 -P，--pollicy

范例 iptables –P INPUT DROP

注解 为链设置缺省的target，通常为ACCEPT和DROP，可以理解为防火墙的缺省策略：除非特定的被运行，其他的都被禁止或除非特定的被禁止，其他的都被允许。

上表列出了一些主要的iptables的命令，此外还有其他的一些命令不很常用，我们在这里不作介绍。这些命令有一些选项，要想详细了解选项的具体内容，可以查看iptables的manpage页。

### 1.8.3 match

在iptables 的一条策略中，如何匹配一个数据包是非常关键的。这一节，我们会详细讨论一些matche，大致可以归为五类。第一类是generic matches（通用的匹配），适用于所有的规则；第二类是TCP matches，顾名思义，这只能用于TCP包；第三类是UDP matches，当然它只能用在UDP包上了；第四类是ICMP matches ，针对ICMP包的；第五类比较特殊，针对的是状态（state），所有者（owner）和访问的频率限制（limit）等。

通用匹配：无论我们使用的是何种协议，也不管我们又装入了匹配的何种扩展，通用匹配都是可用的。也就是说，它们可以直接使用，而不需要什么前提条件，在后面你会看到，有很多匹配操作是需要其他的匹配作为前提的。

-p, –protocol，匹配指定的协议，协议可以用名字来表示，比如tcp、udp、icmp等，名字是部分大小写的，也可以使用他们的整数值，比如tcp 对应的是整数 1、udp对应 17，tcp对应 6。在缺省情况下，如果不写这个匹配，代表所有ALL，但要注意，ALL表示tcp、udp、icmp这三种协议，而不包括/etc/protocol中的所有协议。如果有多个协议需要匹配，可以使用逗号分割，例如：

#iptables –A INPUT –p tcp,udp –j ACCEPT

在协议的前面用“！”标示，代表除了“逻辑非”，例如：

＃iptables –A INPUT –p ！tcp –j DROP

这个表达式代表只允许tcp协议通过，而udp和icmp全部被禁止通过。

-s, --src, --source，匹配数据包的源地址，地址的表示形式如下：

单个地址，如 192.168.1.1，也可写成 192.168.1.1/32 或 192.168.1.1/255.255.255.255

网络地址，如 192.168.1.0，也可写成 192.168.1.0/24 或 192.168.1.0/255.255.255.0

在地址前加“！”表示去反，如 ！192.168.1.0 表示除这个地址段外的所有地址。

如果不在一条策略中注明地址，表示所有地址，也可以表示成 0.0.0.0/0

-d, --dst, –destination，匹配数据包的目的地址，表示方法于源地址的表示方法一致。

-i, --in-interface，以包进入本地所使用的接口来匹配数据包，注意，这个匹配只适用于INPUT、FORWARD、PREROUTING链中，而用在其他任何地方都会出错。可以使用接口的名称来标示数据包的入口，如eth0、ppp0 等，也可以使用通配符，如eth+，表示匹配从所有的以太接口进入的数据包，和前面的一些匹配特性一样，我们可以使用去反符号“！”来标示除了被列出的接口的所有接口。

-o, --out-interface，以包离开本地所使用的接口来匹配数据包，匹配方法和-i的匹配方法完全一致

＃iptables –A FORWORD –i eth0 –o eth1 –p tcp –j ACCEPT

上例说明了凡是从eth0 接口进入，从eth1 接口流出的tcp数据流被允许通过。

隐含匹配：这种匹配操作是自动地或隐含地装载入内核的。例如我们使用参数-p tcp 时，不需再装入任何东西就可以匹配只有IP包才有的一些特点。有三种隐含的匹配针对三种不同的协议，即TCP matches，UDP matches和 ICMP matches。它们分别包括一套只适用于相应协议的判别标准。相对于隐含匹配的是显式匹配，它们必须使用-m或--match被明确地装载，而不能是自动地或隐含地，下一节会介绍到。

tcp matches，该匹配只匹配tcp包的细节，它们必须有-p tcp做为前提条件。主要有以下几种匹配参数：

--sport，--source-port，匹配源端口，有几个原则需要注意：

1、不使用此项，则表示匹配所有端口。

2、可以使用服务名或端口号，使用服务名是为了简化用户的配置，也许用户知道服务名，但不一定知道服务对应的端口号，服务名必须在/etc/services 文件中进行标注，当然简化配置的代价是iptables必须花费额外的系统资源在/etc/services文件中查询服务名对应的端口号，如果用户知道服务名对应的端口号，

我们还是建议用户直接使用端口号来定义。

3、可以使用连续的端口来表示一个服务，比如 –sport 135:139，这表明是从 135 端口到 139 端口，也有这样的表示方法 –sport 1024:，表示匹配源端口从 1024 到 65535。

4、可以在端口号前添加“！”表示除了该端口以为的其他所有端口。

5、不能用这种匹配来标识端口不连续的情况，我们会在后面的部分介绍如何匹配端口不连续的情况。

--dport，--destination-port，匹配目的端口，使用方法与源端口的匹配方法一致。

＃iptables –A FORWARD –p tcp –dport 21:25 –j ACCEPT

上例中的语句表明允许tcp 21 到 25 端口的服务通过防火墙。

udp matches：鉴于udp协议与tcp协议的相似之处：都用端口号来表示一个应用或服务，因此，udp matches的使用方法也和tcp matches的使用方法基本一致，它也有 --sport 、--dport 的匹配原则，但与tcp matches不同的是，udp matches必须与 –p udp配合使用，这是非常容易理解的问题。我们在这里就不需要多说了。

icmp matches：与tcp和udp不同，icmp matches是根据ICMP类型匹配包，类型的指定可以使用十进制数值或相应的名字，数值在RFC792 中有定义，名字可以用iptables --protocol icmp --help 查看。这个匹配也可用英文感叹号取反，如：--icmp-type ！ 8 就表示匹配除类型 8 之外的所有ICMP包。人们经常会使用到icmp协议，但最常用的可能要属icmp echo-request和icmp echo-reply了，也就是我们检测网络连通性的ping命令了。

显示匹配：显示匹配就必须用-m或--match装载，比如要使用状态匹配就必须使用-m state。有些匹配还需要指定协议，有些就不需要，比如连接状态就不要。这些状态是NEW（还未建立好的连接的第一个包），ESTABLISHED（已建立的连接，也就是已经在内核里注册过的），RELATED（由已经存在的、处于已建立状态的连接生成的新连接），等等。有些匹配还处在开发阶段，或者还只是为了说明iptables的强大能力。这说明不是所有的匹配一开始就是实用的，但以后你可能会用到它。随着iptables 新版本的发布，会有一些新的匹配可用。隐含匹配和显式匹配最大的区别就是一个是跟随协议匹配自动装载的，一个是显式装载的。

limit匹配：这个匹配操作必须由-m limit明确指定才能使用。有了它的帮助，就可以对指定的规则的日志数量加以限制，以免系统记录大量重复的日志信息。比如，你可以事先设定一个限定值，当符合条件的包的数量不超过它时，就记录；超过了，就不记录。我们可以控制某条规则在一段时间内的匹配次数（也就是可以匹配的包的数量），这样就能够减少 DoS syn flood攻击的影响。

--limit-burst：这个参数定义了当前策略的峰值，也就是单位时间内匹配的数据包的最大数量，每匹配一个，数值就减一，直到 0 为止，新来的数据包将不被进行匹配操作，也可以将这一数值理解为允许建立连接的阀值。

--limit：这个参数定义了一个相当于频率的概念，在—limit-burst这个匹配项中我们已经提到，当 --limit-burst定义的数值被耗尽，在单位时间内是会得到适当的补充的，这个频率和数量就是在 --limit中定义的。

举个例子也许更容易帮你理解 –limit-burst和 –limit组合使用的功效。具有再充值功能的电话卡（比如神州行卡）想必许多人都用过，设想这样一种情况，某个人在新买的手机上一次性的充了 300 元钱的话费，然后以后每月月底都往卡里充 100 元的话费，但有两个前提：

1、充值后的卡内余额不得超过 300 元。

2、当月能且只能充 100 元的话费。即便最初的 300 元话费全部用光了，也只能补充这么多的话费。

在上例中我们可以将那个 300 元话费看成 --limit-burst 定义的数值，将每月的 100 元话费看成 --limit定义的数值，记住 --limit 是有单位的，在iptables中，这个单位可以是second、min、hour、day。

我们可以用limit功能来做一些防止Dos攻击的工作，看下面的例子：

＃iptables –A INPUT –p tcp --dport 80 –m limit --limit 10/second --limit-burst 200 –j ACCEPT

上面的例子是在传达这样一个意思，主机的 80 端口对外开放，系统资源允许每秒新建立 250 个会话，主机通过iptables设置了能够提供每秒 200 个新会话的容量（通常情况下，我们需要为服务器或主机本身考虑一些冗余），在正常情况下，这个数值完全能够满足应用的要求，但如果有某个攻击者对主机的 80 端口进行拒绝服务攻击，每秒 200 个新会话的容量也许一会的功夫就被用光了，对于一个没有保护措施的主机来讲，系统马上就会瘫痪，但由于我们设置了一定的保护措施，即便有攻击过来，在iptables这一关就会被丢弃。在这之后，iptables会每秒重新给主机 10 个新的会话名额，使之能够处理新的连接。在新来的连接请求中，有正常的连接请求，也有一些是攻击，但无论如何，通过这种方式，我们保证了服务器始终不会被攻瘫掉，在更有效的对付拒绝服务攻击的新技术出现以前，这也许是最有效的方法。

MAC匹配：基于包的MAC源地址匹配包，地址格式只能是XX:XX:XX:XX:XX:XX，当然它也可以用英文感叹号取反，如--mac- source ！00:00:00:00:00:01，意思很简单了，就是除此之外的地址都可接受。注意，因为 MAC addresses只用于Ethernet类型的网络，所以这个match只能用于Ethernet接口。而且，它还只能在PREROUTING， FORWARD 和INPUT链里使用。

MAC匹配有一个非常好的应用就是可以进行IP地址和MAC地址的绑定，对于一些安全要求较高的网络，这是一个有效的配置策略。

＃iptables –A FORWARD –s 192.168.1.23 –m mac –mac-source 00:e0:4c:3d:5e:4f –j ACCEPT

上例表明源IP地址为 192.168.1.23，源MAC地址为 00:e0:4c:3d:5e:4f的计算机能够通过防火墙，如果这个用户将自己的IP地址改成了 192.168.1.24，而该地址同样也做了MAC地址的match，那么要想通过更改IP地址而获取上网权限的企图是徒劳的。

说明：可以通过iptables实现IP地址与MAC地址绑定的功能，但还有一种更加高效的方法，我们在这里简要的说明一下，在/etc/目录下编辑文件ethers，在文件中添加如下内容：

192.168.1.1 00:e0:4c:3d:5e:4f

192.168.1.1 00:e0:4c:3d:5b:3d

…….

文件编辑完成后执行命令：

＃/sbin/arp –f

state匹配：state匹配在防火墙配置过程中非常重要的，如果没有state的配置，那么需要配置双向的规则才能满足通讯的需要，但防火墙既然有状态检测功能，我们为什么不好好使用它呢？

--state：state的状态有四种，指定要匹配包的的状态，当前有 4 种状态可用：INVALID，ESTABLISHED，NEW和RELATED。四种数据包的状态我们在前面已经做了详细的描述本节我们只进行state的用法描述，如下例所示：

＃iptables –A FORWARD –p tcp –m state --state RELATED,ESTABLISHED –j ACCEPT

本例表明凡是数据包状态为“RELATED”、“ESTABLISHED”的tcp包允许通过防火墙，在一般情况下，基于状态的策略都是配置在每一条 chain的最前面，因为当包被匹配到以后，就能够直接被处理了，这是一种比较高效的配置方法。关键字ESTABLISHED比较容易理解，即匹配状态为 “已经建立连接”的数据包，那么怎么理解“RELATED”呢，RELATED表示不属于已经建立的那条连接，但和那条连接有关，比如ftp，ftp在建立连接的过程中会首先建立一条ftp-control连接用以传输指令等，真正传输数据的是一条叫做ftp-data的连接，而传输数据的连接是和传输控制信号的连接相关的，因此“RELATED”是用于类似这些特殊服务的。在正常情况下，对于每一种协议：TCP、UDP、ICMP都可以单独的配置状态策略，但一种比较简单高效的做法是：

＃iptables –A INPUT –p all –m state --state RELATED,ESTABLISHED –j ACCEPT

multiport:这个匹配选项为我们解决了如何在一条策略种匹配那些端口不连续的服务，在一般情况下，一个公司或企业的安全策略是允许内部网络使用有限的Internet服务，如收发电子邮件、上网浏览网页、msn聊天等，看看下面的例子：

#iptables –A FORWARD –i eth0 –p tcp –m multiport --dports 25,80,110,443,1863 –j ACCEPT

#iptables –A FORWARD –i eth0 –p udp --dport 53 –j ACCEPT

仅仅两条命令就解决了内部用户上网收发E_mail、浏览网页、使用msn聊天等需求，怎么样，就这么简单。

1.8.4 target和jump

从字面的意思来理解，target表示目标，jump表示跳转，两个结合起来表示被匹配到的数据包将跳转的哪个目标去，并执行那个目标相应的动作，DROP、ACCEPT或其他。还是举个例子有助于我们理解target和jump。

#iptables –N power_users

#iptables –A power_users –p all –j ACCEPT

#iptables –A FORWARD –s 192.168.1.0 –j power_users

上面的例子中，第一句表示新建一条chain，名称叫做power_users，第二句是对power_users链定义策略，该策略表明接受所有协议的数据包，确实如此，否则我们也不会将这条链成为power_users了，第三条语句定义了一个转发策略，将原地址为 192.168.1.0/24 网段的用户产生的数据包被跳转至power_users链处理。

在本节中，我们将对比较常用的几个targets进行详细的描述，他们分别是DROP、ACCEPT、SNAT、DNAT、MASQUERADE、LOG。

DROP： DROP target表示被匹配的数据包将要被执行的动作是丢弃，也就是说，在这里，包的生命走到了终点，在这种情况下，iptables对包的处理只是将包丢弃，它不会对包的发送者返回任何信息，也不会向路由器发送任何错误信息，这在某些情况下会造成一些问题，比如发送者还在苦苦等待回包的到来，而发送出去的包早就在半路上夭折了，可以通过REJECT 9target来避免这个问题，REJECT和DROP的差别在于它不仅仅丢弃包，而且还礼节性的告诉发送者，这样就不至于发送者苦苦等待了。其实对于一般的应用，我们并不建议使用REJECT，既然包总是要被DROP的，那就让发送者去等吧，这和我们有什么关系呢？

ACCEPT：这个target没有任何选项和参数，使用也很简单，指定-j ACCEPT即可。一旦包满足了指定的匹配条件，就会被ACCEPT，并且不会再去匹配当前链中的其他规则或同一个表内的其他规则。

SNAT：这个target是用来做源网络地址转换的，就是重写包的源IP地址。当我们有若干台计算机共享一个Internet 连接时，就能用到它了，而且这也是绝大多数企业用户所采用的Internet互联方案。先在内核里打开ip转发功能，然后再写一个SNAT规则，就可以把所有从本地网络出去的包的源地址改为Internet连接的地址了。如果我们不这样做而是直接转发本地网的数据包的话，Internet上的机子就不知道往哪儿发送应答了，因为在本地网里我们一般使用的是IANA组织专门指定的一段地址，它们是不能在Internet上使用的，还记得那些地址范围吗？让我们来回忆一下：

10.0.0.0 - 10.255.255.255 (10/8 prefix)

172.16.0.0 - 172.31.255.255 (172.16/12 prefix)

192.168.0.0 - 192.168.255.255 (192.168/16 prefix)

SNAT target的作用就是让所有从本地网出发的包看起来都是从一台机子发出的，这台机子一般就是防火墙或路由器之类的三层设备。

SNAT只能用在nat表的POSTROUTING链里。只要连接的第一个符合条件的包被SNAT了，那么这个连接的其他所有的包都会自动地被SNAT,而且这个规则还会应用于这个连接所在流的所有数据包。

既然提到了ip包转发的问题，我们也不妨费一些文字做一些简要的说明，当iptables被用来安装在一个提供WWW或E_mail服务的主机上以期为主机提供保护时是不涉及包转发的，但当iptables被用在网络的边界节点充当一个安全网关时，就涉及到包转发的问题了，这时，iptables需要通过不同的网卡来转发数据包，在linux系统中，控制是否允许转发数据包是有一个开关的，0 表示不转发，1 表示转发，只要更改 /etc/sysct.confl文件中的如下内容：

#Controls IP Packets Forwarding

net.ipv4.ip_forward = 1

然后重启系统，这时系统就能够进行包转发了，还有一种方法是在内核空间内直接更改，通过改下列文件的参数：

#echo 1 > /proc/sys/net/ipv4/ip_forward

注意，这个文件一旦更改，立即生效，但如果重启系统，将回到默认设置，所以/etc/sysctl.conf文件无论如何都应该将转发开关置位为“1”。

看看下面的范例，一般一个SNAT的命令是这样完成的：

#iptables –t nat –A POSTROUTING –s 192.168.1.0 –j SNAT --to-source 1.1.1.1

这条命令表示将内部网络的 192.168.1.0/24 网段的地址翻译成 1.1.1.1，这样内部地址就可以使用一个公有IP（Public IP address）地址共享上网了，当然，还要有额外的策略允许内部用户上网才行哦，别忘了。

SNAT只有一个参数，--to-source，它有几种使用方法：

1、单独的地址，就象上面的例子所示。

2、一段连续的地址，用连字符分隔，如 1.1.1.1-1.1.1.10，这样可以实现负载平衡。每个流会被随机分配一个IP，但对于同一个流使用的是同一个IP。当然，前提是你要有这么多的公有ip才行啊！

3、在指定-p tcp 或 -p udp的前提下，可以指定源端口的范围，如 1.1.1.1:1024-32000，这样包的源端口就被限制在 1024-32000 了。

DNAT：这个target是用来做目的网络地址转换的，意思是重写包的目的IP地址。如果一个包被匹配了，那么和它属于同一个流的所有的包都会被自动转换，然后就可以被路由到正确的主机或网络。DNAT target是非常有用的。比如，你的Web服务器在LAN内部，而且没有可在Internet上使用的真实IP地址，那就可以使用这个 target让防火墙把所有到它自己HTTP端口的包转发给LAN内部真正的Web服务器。目的地址也可以是一个范围，这样的话，DNAT会为每一个流随机分配一个地址。因此，我们可以用这个target做某种类型的负载平衡。

DNAT也和SNAT一样只有一个参数，--to-destination，看看下面的例子：

#iptables –t nat –A PREROUTING –d 1.1.1.1 –j DNAT --to-destination 192.168.1.24

上面的例子说明凡是访问 1.1.1.1 的数据包都转给内网的 192.168.1.24，这样内部的服务器就可以发布出去了。现在问题来了，如果仅仅这样配置还是存在一点问题的，假设内部网络的其他用户也访问这个经过映射后的地址会怎样？当然他们完全可以直接访问服务器的真实地址，但在一些具体的情况下，比如 dns的解析就是指向了映射后的地址。可以加一条这样的语句：

#iptables –t nat –A POSTROUTING –d 1.1.1.1 –j SNAT –to-source 192.168.1.254

在这里，假设防火墙的内网地址是 192.168.1.254，通过这样的配置，问题解决了，想一想为什么会这样？看看下面的示意图：也许你就明白了：

知道问题出在哪里了吗？很明显，当客户机 192.168.1.23 发起一个访问请求给映射后的地址 1.1.1.1，防火墙收到这个请求后根据策略表匹配发现是一个对内部服务器 192.168.1.24 的映射，如果不加上面的那一行语句，防火墙会通过纯路由的方式将数据包转发给服务器 192.168.1.24，服务器收到请求后，发现源地址为 192.168.1.23 的客户机发来了一个请求，并且这台主机与自己在同一个网段内，于是直接将回应包SYN+ACK发送给主机 192.168.1.23，主机收到这个包后会感觉很奇怪，因为它从来就没有给 192.168.1.24 发送过连接请求报文，所以就会将回应报文丢弃，现在，我们对内网主机通过映射后地址访问内网服务器的需求进行一些修改，如上面的哪条命令，我们对所有到 1.1.1.1 的连接都做一个原地址路由，将连接的原地址改变成防火墙的内网接口地址，于是问题解决了。

当然，问题总有两面性，对于服务器而言，所有访问者这时都变成了防火墙内网的接口地址，这对于服务器的审计是有影响的，在具体的配置过程中，使用者需要权衡利弊，在做配置决定。

MASQUERADE：这个链与SNAT的差别不大，回忆一下在进行SNAT的配置时需要指明一个固定的映射地址，但如果用户的网络使用的是ADSL这种动态获取IP地址的方式，SNAT就不适用了，这时只能使用MASQUERADE（地址伪装），在具体的处理过程中，系统需要读取当前的动态地址，然后用当前的地址对数据包进行重新封装：

＃iptables –A POSTROUTING –o eth1 –s 192.168.1.0/24 –j MASQUERADE

MASQUERADE可以添加参数--to-ports 1024-30000，可以通过这个参数指定映射后的源端口范围。

LOG：顾名思义，这条链是用来记录日志的。记录日志有助于必要是为我们提供一些有用的信息，比如被防火墙DROP的数据包就有可能包含一些危险的动作，网络发生故障时，我们也能够通过LOG发现一些潜在的问题。

有两个常用的选项，--log-level 、--log-prefix

--log-level 告诉iptables和 syslog使用哪个记录等级。记录等级的详细信息可以查看文件syslog.conf，一般来说有以下几种，它们的级别依次是：debug，info， notice，warning，warn，err，error，crit，alert， emerg，panic。其中，error和err、warn和warning、panic和emerg分别是同义词，也就是说作用完全一样的。注意这三种级别是不被赞成使用的，换句话说，就是不要使用它们（因为信息量太大）。信息级别说明了被记录信息所反映的问题的严重程度。所有信息都是通过内核的功能被记录的，也就是说，先在文件 syslog.conf里设置

kern.info /var/log/iptables

然后再让所有关于iptables的LOG信息使用级别info，就可以把所有的信息存入文件/var/log/iptables内。注意，其中也可能会有其他的信息，它们是内核中使用info 这个等级的其他部分产生的。有关日志的详细信息，可以syslog和syslog.conf的man page。

--log-prefix 描述日志的前缀，这样和grep或其他工具一起使用时就容易追踪特定的问题，而且也方便从不同的规则输出。前缀最多能有 29 个英文字符，这已经是包括空白字符和其他特殊符号的总长度了。

＃iptables –A FORWARD –p tcp –j LOG –log-prefix “tcp_packets “ –log-level info

这条语句表示需要记录所有tcp协议的数据包，并以“tcp_packets”做为标记。

第二章 范例

网络环境



如上所示：这是一个非常典型的企业网络拓扑结构，通过固定IP地址接入Internet，内部网络通过防火墙与Internet互连并进行安全控制，只允许内部网络的用户通过ftp、http、smtp、和pop3 与Internet连接，而禁止所有从外网到内网的连接，内部用户通过防火墙的外网接口地址共享上网。运行脚本firewall.sh如下：

```
#!/bin/sh
# example.firewall - Initial SIMPLE IP Firewall script for Linux 2.4.x and iptables
##########################################################################
# Configuration options.
# Internet Configuration.
INET_IP="194.236.50.155"
INET_SERVER=”194.236.50.156”
INET_IFACE="eth0"
INET_BROADCAST="194.236.50.255"
##########################################################################
#Lan configuration
```

```
# your LAN's IP range and localhost IP. /24 means to only use the first 24
# bits of the 32 bit IP address. the same as netmask 255.255.255.0
LAN_IP="192.168.0.254"
LAN_SERVER=”192.168.0.1”
LAN_IP_RANGE="192.168.0.0/24"
LAN_BROADCAST=”192.168.0.255”
LAN_IFACE="eth1"
############################################################
####################
# Localhost Configuration.
LO_IFACE="lo"
LO_IP="127.0.0.1"
############################################################
####################
# IPTables Configuration.
IPTABLES="/usr/sbin/iptables"
############################################################
####################
# Module loading.
# Needed to initially load modules
/sbin/depmod -a
# Required modules
/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit
/sbin/modprobe ipt_state
############################################################
####################
# Non-Required modules
#/sbin/modprobe ipt_owner
#/sbin/modprobe ipt_REJECT
#/sbin/modprobe ipt_MASQUERADE
/sbin/modprobe ip_conntrack_ftp
#/sbin/modprobe ip_conntrack_irc
/sbin/modprobe ip_nat_ftp
############################################################
####################
# /proc set up.
# Required proc configuration
echo "1" > /proc/sys/net/ipv4/ip_forward
############################################################
```

```
####################
# rules set up.
# Filter table
# Set default policies
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP
############################################################
####################
# Create userspecified chains
# Create chain for bad tcp packets
$IPTABLES -N bad_tcp_packets
# Create separate chains for ICMP, TCP and UDP to traverse
$IPTABLES -N allowed
$IPTABLES -N tcp_packets
$IPTABLES -N udp_packets
$IPTABLES -N icmp_packets
############################################################
####################
# Create content in userspecified chains
# bad_tcp_packets chain
$IPTABLES -A bad_tcp_packets -p tcp --tcp-flags SYN,ACK SYN,ACK \
-m state --state NEW -j REJECT --reject-with tcp-reset
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "New not syn:"
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP
############################################################
####################
# allowed chain
$IPTABLES -A allowed -p TCP --syn -j ACCEPT
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A allowed -p TCP -j DROP
############################################################
####################
# TCP rules for the services of ftp, ssh, smtp,http,pop3
#
$IPTABLES -A tcp_packets -p TCP –m multiport --dports 21,22,25,80,110 -j allowed
############################################################
####################
# UDP rules for the services of dns
#
$IPTABLES -A udp_packets -p UDP --destination-port 53 -j ACCEPT
#
# In Microsoft Networks you will be swamped by broadcasts. These lines
```

```
# will prevent them from showing up in the logs. Uncomment the following
# line to make the policy active if you have such a network
#
#$IPTABLES -A udp_packets -p UDP -i $INET_IFACE -d $INET_BROADCAST \
#--destination-port 135:139 -j DROP
#$IPTABLES -A udp_packets -p UDP -i $LAN_IFACE -d $LAN_BROADCAST \
#--destination-port 135:139 -j DROP
##########################################################
####################
# ICMP rules
#
$IPTABLES -A icmp_packets -p ICMP --icmp-type 8 -j ACCEPT
##########################################################
####################
# INPUT chain
#
# Bad TCP packets you don't want.
#
$IPTABLES -A INPUT -p tcp -j bad_tcp_packets
#
# Rules for special networks not part of the Internet
#
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -s $LAN_IP_RANGE -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT
# Drop the packet from Internet
#
$IPTABLES -A INPUT -p ALL -i $INET_IFACE –j DROP
# Log weird packets that don't match the above.
#
$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix " INPUT packet died: "
##########################################################
####################
# FORWARD chain
#
# Bad TCP packets we don't want
#
$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets
#
# Accept the packets we actually want to forward
#
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
$IPTABLES -A FORWARD –p tcp -i $LAN_IFACE -j tcp_packets
$IPTABLES -A FORWARD –p udp -i $LAN_IFACE -j udp_packets
$IPTABLES -A FORWARD -p icmp -i $LAN_IFACE -j icmp_packets
#
# Log weird packets that don't match the above.
#
$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "FORWARD packet died: "
#
# OUTPUT chain
#
# Bad TCP packets we don't want.
#
$IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets
#
# Special OUTPUT rules to decide which IP's to allow.
#
$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT
#
# Log weird packets that don't match the above.
#
$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "OUTPUT packet died: "
##########################################################
####################
# nat table
-j DNAT --to-destination $LAN_SERVER
# POSTROUTING chain
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source $INET_IP
```

编辑好脚本后，为其添加可执行权限：

#chmod ugo+x firewall.sh

将firewall.sh脚本移动到/etc目录下：

#mv firewall.sh /etc/

确保系统在启动是能够运行该脚本，添加如下语句到/etc/rc.d/rc.local文件中：

#echo sh /etc/firewall.sh >> /etc/rc.d/rc.local

第三章 Iptables的新功能

在netfilter .9的官方网站，2004 年 03 月 02 日，发布了新的patch，在新的patch中公布了一些iptables可以使用的新的功能，有一些还是非常实用的。比如可以用来定义IP地址范围段的iprange、定义连续或不连续端口范围的mport、定义时间规则的time，定义网络流量配额的quota等，这些功能本人已经测试过，基本上还不错，但其他的一些功能，在本人来看还不是特别实用，因此没有测试，读者如果感兴趣，也可以自己用用看，别忘了告诉我结果哦

在使用这些新的功能前，首先需要将这些patch编译进核心，当然也可以将他们做为Modules使用，至于

如何编译核心，我在前面已经讲过了。

写到这里，花了我好几天的功夫，实在是累了，这一章就不仔细写了，就好像维纳斯缺一个胳膊却是美的化身，本文也在第三章留一些缺憾，相信如果您已经仔细的阅读了本文的前两章，并进行了实际的操作，相信在第三章，您可以凭借自己的能力搞定我没有仔细阐述的那些问题的。