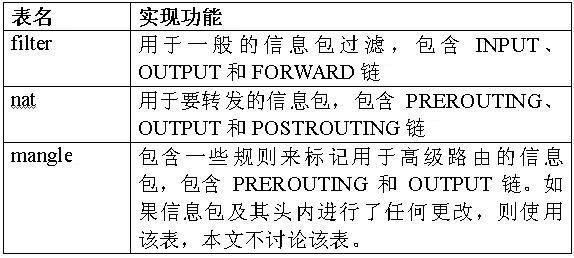
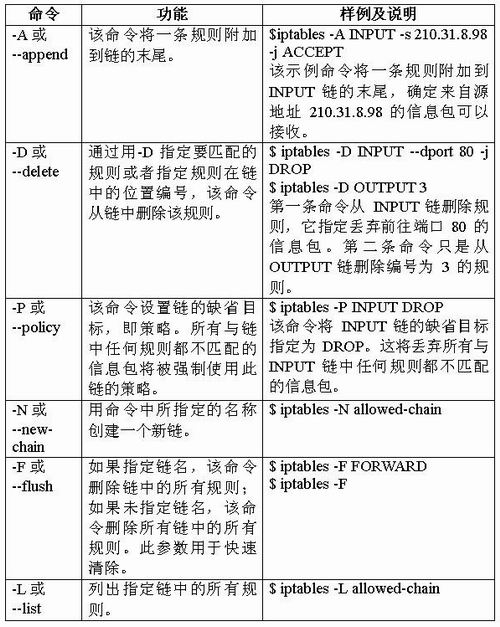
**防火墙配置文件iptables详解**

对于Internet上的系统，不管是什么情况都要明确一点：网络是不安全的。因此，虽然创建一个防火墙并不能保证系统100％安全，但却是绝对必要的。 Linux提供了一个非常优秀的防火墙工具—netfilter/iptables。它完全免费、功能强大、使用灵活、可以对流入和流出的信息进行细化控 制，且可以在一台低配置机器上很好地运行。本文将简单介绍使用netfilter/iptables实现防火墙架设和Internet连接共享等应用。

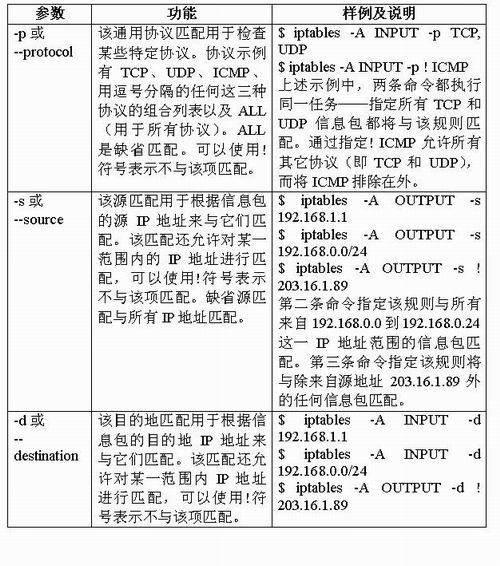
netfilter/iptabels应用程序，被认为是 Linux中实现包过滤功能的第四代应用程序。netfilter/iptables包含在2.4以后的内核中，它可以实现防火墙、NAT（网络地址翻 译）和数据包的分割等功能。netfilter工作在内核内部，而iptables则是让用户定义规则集的表结构。netfilter/iptables 从ipchains和ipwadfm（IP防火墙管理）演化而来，功能更加强大。下文将netfilter/iptabels统一称为iptables。

可以用iptables为Unix、Linux和BSD个人工作 站创建一个防火墙，也可以为一个子网创建防火墙以保护其它的系统平台。iptales只读取数据包头，不会给信息流增加负担，也无需进行验证。要想获得更 好的安全性，可以将其和一个代理服务器（比如squid）相结合。

基本概念   
典型的防火墙设置有两个网卡：一个流入，一个流出。iptables读取流入和流出数据包的报头，将它们与规则集（Ruleset）相比较，将可接受的数据包从一个网卡转发至另一个网卡，对被拒绝的数据包，可以丢弃或按照所定义的方式来处理。   
通过向防火墙提供有关对来自某个源地址、到某个目的地或具有特定协议类型的信息包要做些什么的指令，规则控制信息包的过滤。通过使用iptables系统 提供的特殊命令iptables建立这些规则，并将其添加到内核空间特定信息包过滤表内的链中。关于添加、去除、编辑规则的命令，一般语法如下：   
iptables [-t table] command [match] [target]

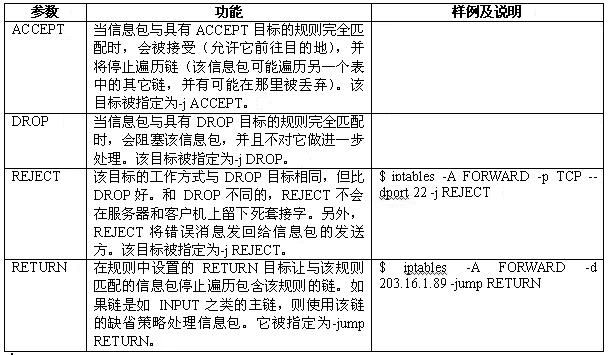
1．表（table）   
[-t table]选项允许使用标准表之外的任何表。表是包含仅处理特定类型信息包的规则和链的信息包过滤表。有三个可用的表选项：filter、nat和mangle。该选项不是必需的，如果未指定，则filter作为缺省表。各表实现的功能如表1所示。  
表1 三种表实现的功能  
  
<!--[if !vml]-->2．命令（command）   
command部分是iptables命令最重要的部分。它告诉iptables命令要做什么，例如插入规则、将规则添加到链的末尾或删除规则。表2是最常用的一些命令及例子。   
表2 命令的功能和样例  
  
<!--[if !vml]--><!--[endif]-->  
3．匹配（match）

iptables命令的可选match部分指定信息包与规则匹配所应具有的特征（如源地址、目的地址、协议等）。匹配分为通用匹配和特定于协议的匹配两大类。这里将介绍可用于采用任何协议的信息包的通用匹配。表3是一些重要且常用的通用匹配及示例说明。   
表3 通用匹配及示例说明

  
  
<!--[if !vml]--><!--[endif]-->  
4．目标（target）

目标是由规则指定的操作，对与那些规则匹配的信息包执行这些操作。除了允许用户定义的目标之外，还有许多可用的目标选项。表4是常用的一些目标及示例说明。

除表4外，还有许多用于建立高级规则的其它目标，如LOG、REDIRECT、MARK、MIRROR和MASQUERADE等。

表4 目标及示例说明  
  
  
应用iptables

与ipchains和ipfwadm不同的是，iptables可以配置有状态的防火墙。iptables可以检测到源地址和目的地址、源端口和目 的端口及流入数据包的顺序，即iptables记住了在现有连接中，哪些数据包已经被允许接收。这使得暂时性的端口只有在需要时才会被打开，并且会拒绝所 有永久性占用端口的请求，大大地加强了安全性。同时，那些被更改了报头的数据包，即使包含有一个被允许的目的地址和端口，也会被检测到并被丢弃。此外，有 状态的防火墙能够指定并记住为发送或接收信息包所建立连接的状态。防火墙可以从信息包的连接跟踪状态获得该信息。在决定新的信息包过滤时，防火墙所使用的 这些状态信息可以增加其效率和速度。

1．启动和停止iptables

下面将正式使用iptables来创建防火墙。启动和停止iptables的方法取决于所使用的Linux发行版，可以先查看所使用Linux版本的文档。

一般情况下，iptables已经包含在Linux发行版中，运行iptables --version来查看系统是否安装了iptables。在Red Hat 9.0中，安装的版本是iptables v1.2.7a。如果系统没有安装iptables，则可以从http://www.netfilter.org下载。

2．查看规则集

上面仅对iptables的用法做了一个简单介绍，使用中可以运行man iptables来查看所有命令和选项的完整介绍，或者运行iptables -help来查看一个快速帮助。要查看系统中现有的iptables规划集，可以运行以下命令：

iptables --list

下面是没有定义规划时iptables的样子：

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   
如上例所示，每一个数据包都要通过三个内建的链（INPUT、OUTPUT和FORWARD）中的一个。

filter是最常用的表，在filter表中最常用的三个目标是ACCEPT、DROP和REJECT。DROP会丢弃数据包，不再对其进行任何处理。REJECT会把出错信息传送至发送数据包的主机。   
<!--[if !vml]--><!--[endif]-->   
在Red Hat 9.0中，提供一个GUI程序来让用户对系统的安装级别进行简单的配置。该工具的启动方法是：主选单→系统设置→安全工具（如图1所示）。在此将安全级别 设为“高级”，并选择使用默认的防火墙规则。点击确定后，再用iptables -list显示，发现iptables与没有定义规则前已经有很大不同，如下所示：

[root@workstation root]# iptables --list   
Chain INPUT (policy ACCEPT)   
target prot opt source destination   
RH-Lokkit-0-50-INPUT all -- anywhere anywhere

Chain FORWARD (policy ACCEPT)   
target prot opt source destination   
RH-Lokkit-0-50-INPUT all -- anywhere anywhere

Chain OUTPUT (policy ACCEPT)   
target prot opt source destination   
......

现实中一般不使用这个GUI工具，因为它的功能有限，也不够透明。相比较而言，SuSE 9.0中相应的配置工具要好得多，它可以在GUI下对防火墙进行更加细化的配置（比如增加了IP转发和伪装等功能的配置）。尽管这样，一般还是自己来增加和删除规则。

．增加规则

本例中的规则将会阻止来自某一特定IP范围内的数据包，因为该IP地址范围被管理员怀疑有大量恶意攻击者在活动：

# iptables -t filter -A INPUT -s 123.456.789.0/24 -j DROP

也可以很轻易地阻止所有流向攻击者IP地址的数据包，该命令稍有不同：

# iptables -t filter -A OUTPUT -d 123.456.789.0/24 -j DROP

注意这里的A选项，如前所述，使用它说明是给现有的链添加规则。

4．删除规则

网络上的恶意攻击者总是在变化的，因此需要不断改变IP。假设一个网上攻击者转移到新的IP地址，而其老的IP地址被分配给一些清白的用户，那么这时这些用户的数据包将无法通过你的网络。这种情况下，可以使用带-D选项的命令来删除现有的规则：

# iptables -t filter -D OUTPUT -d 123.456.789.0/24 -j DROP

5．缺省的策略

创建一个具有很好灵活性、可以抵御各种意外事件的规则需要大量的时间。对于那些没有时间这样做的人，最基本的原则是“先拒绝所有的数据包，然后再允许需要的”。下面来为每一个链设置缺省的规则：

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

这里选项-P用于设置链的策略，只有三个内建的链才有策略。这些策略可以让信息毫无限制地流出，但不允许信息流入。很多时候需要接收外部信息，则可使用以下命令：

# iptables -t filter -A INPUT -s 123.456.789.0/24 -j ACCEPT

6．SYN的使用

不能关闭所有端口，也不能只指定某些端口处于打开状态，那么怎样才能设置一个有效的规则，既可以允许普通用户正常通过，又可以阻止恶意攻击者访问网络呢？

刚开始使用iptables的人可以充分利用syn标识来阻止那些未经授权的访问。iptables只检测数据包的报头，事实上，除 iptables以外，很多其它有用的数据包分析都是基于报头的。比如，在进行Web冲浪时，一个请求从你的PC发送至其它地方的Web服务器上，该服务 器会响应请求并发回一个数据包，同时得到你系统上的一个临时端口。与响应请求不同的是，服务器并不关心所传送的内容。可以利用这种特点来设置规则，让它阻 止所有没有经过你系统授权的TCP连接：

# iptables -t filter -A INPUT -i eth0 -p tcp --syn -j DROP

这里的-i指的是网卡，-p则是指协议，--syn则表示带有syn标识设置的TCP数据包。SYN用于初始化一个TCP连接，如果自己机器上没有运行任何服务器，别人也就不会向你发送SYN数据包。

7．有状态的数据包的检测

前边的例子把每一个数据包看成是独立的，而不是相互关联的，依靠的是数据包的头信息。iptables会检查数据包的源和目的IP地址、源和目的端 口、流入数据包的顺序号、TCP先后顺序的信息及头标记（SYN、ACK、FIN、RST等）的状态，即它会跟踪整个连接会话，从而使整个过滤过程是相互 关联的。

8．共享一个Internet连接

网络地址翻译和IP伪装都可以实现多台主机共享一个Internet连接，这个局域网可以是Linux和Windows系统组成的多系统局域网。假 设现在有一台机器，配有两个网卡，其中eth0为“公共”网卡，eth1为“私有”网卡，即eth0被分配了一个静态的、可路由的IP地址，而eth1被 分配了一个私有的、不能路由的IP，该IP是属于该局域网子网的。要实现上述功能，需要向nat和filter表中添加一些链：

# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE   
# iptables -t filter -A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT   
# iptables -t filter -A FORWARD -i eth1 -o eth0 -j ACCEPT

这显示了有状态的数据包检测的价值。请注意，这里是如何实现流入数据包只有在属于一个已经存在的连接时才被允许，而所有来自局域网内流向外的数据包则都允许通过。第一条规则让所有流出的信息看起来都是来自防火墙机器的，而并不会显示出防火墙后面还有一个局域网。

下面的命令为FORWARD和POSTROUTING链设置缺省的策略，在使用伪装时，有一个缺省的POSTROUTING DROP策略非常重要，否则就可能有心怀恶意的用户突破网关后伪装自己的身份。

# iptables -t filter -P FORWARD DROP   
# iptables -t nat -P POSTROUTING DROP

下面的命令为拨号连接设置，它可以动态地分配IP地址：

# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

9．运行服务器时的情况

有时也会把服务器放置在防火墙后面，这时iptables就需要知道从哪儿通过数据包，设置如下所示：

# iptables -t nat -A PREROUTING -i eth0 -p tcp -dport 80 -j DNAT -to 192.168.0.10:80   
# iptables -t nat -A PREROUTING -i eth0 -p tcp -dport 25 -j DNAT -to 192.168.0.11:25

10．规则的保存

到现在为止，所有的例子都是在命令行中进行的。在测试新的规则时，这是一种很好的方式，但一旦测试结果令人满意，就可以将它们保存为脚本。可以使用 iptables-save 命令来实现：

$ iptables-save >; iptables-script

信息包过滤表中的所有规则都被保存在文件iptables-script中。无论何时再次引导系统，都可以使用iptables-restore命令将规则集从该脚本文件恢复到信息包过滤表。恢复命令如下所示：

$ iptables-restore iptables-script

如果愿意在每次引导系统时自动恢复该规则集，则可以将上面指定的这条命令放到任何一个初始化Shell脚本中。

下面的例子并不是一个完整的脚本，它只是描述了如何使用变量及提供了一些附加的规则样例。

#!/bin/sh   
#为变量赋值   
IPTABLES=/sbin/iptables   
LAN\_NET="192.168.1.0/24"   
IFACE= "eth0"   
LO\_IFACE="lo"   
LO\_IP="127.0.0.1"   
#加载所需的内核   
/sbin/modprobe ip\_conntrack   
/sbin/modprobe iptable\_nat   
#缺省情况下，IP转发都处于不可用状态，将其设置为可用状态：   
echo "1" >; /proc/sys/net/ipv4/ip\_forward   
#使IP的动态分配功能可用   
echo "1" >; /proc/sys/net/ipv4/ip\_dynaddr   
#每次重启这个脚本时，最好清除以前所设的规则   
$IPTABLES -P INPUT DROP   
$IPTABLES -F INPUT   
$IPTABLES -P OUTPUT ACCEPT   
$IPTABLES -F OUTPUT   
$IPTABLES -P FORWARD DROP   
$IPTABLES -F FORWARD   
$IPTABLES -F -t nat   
#只允许在LAN中使用SSH连接   
$IPTABLES -A INPUT -s LAN\_NET -p tcp --destination-port ssh -j ACCEPT   
#允许loopback!   
$IPTABLES -A INPUT -i lo -p all -j ACCEPT   
$IPTABLES -A OUTPUT -o lo -p all -j ACCEPT   
#丢弃那些流入的宣称是来自本地机器的数据包   
#丢弃那些流出的不是出自本地机的数据包   
$IPTABLES -A INPUT -i $IFACE -s $LAN\_NET -j DROP   
$IPTABLES -A OUTPUT -o $IFACE -s ! $LAN\_NET -j DROP   
#限制一些流出的信息   
$IPTABLES -A OUTPUT -o eth0 -p tcp -dport 31337 -j DROP   
$IPTABLES -A OUTPUT -o eth0 -p tcp -sport 31337 -j DROP   
#此外，31335、27444、27665、20034 NetBus、9704、137-139（smb）端口也应被禁止。