目录

[linux之iptables详解及配置（一） 1](#_Toc370124383)

[linux之iptables详解及配置（二）+案例 15](#_Toc370124384)

[iptables之mangle表应用实现策略路由+（案例） 23](#_Toc370124385)

# linux之iptables详解及配置（一）

在Internet中，企业通过架设各种服务器系统为用户提供丰富多彩的网络应用，例如Web网站、电子邮件系统、FTP服务器、数据库系统等。那么如何来保护这些特定的服务，过滤企业不需要的访问甚至是恶意的入侵呢？使用防火墙正是强有力的防护措施之一。

1、Linux防火墙基础

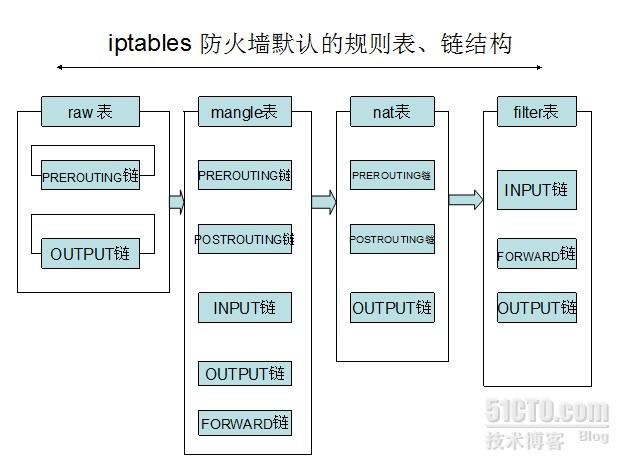
作为隔离内外网、过滤非法数据的有力屏障，防火墙通常按实现环境的不同分为硬件防火墙和软件防火墙。硬件防火墙是功能专一的硬件设备，具有比较全面的功能，其工作效率较高，但是加个昂贵，通常只应用于非常重要的主干网络节点。而软件防火墙的功能是由操作系统或软件程序实现的，可以在Linux或者Windows等系统平台构建软件防火墙。软件防火墙的价格优势比较明显，配置也相对灵活，如果设置得当，同样可以实现硬件防火墙具有的功能和效率。Internet中有大量的企业网络在使用Linux系统搭建软件防火墙

1.1>、Linux的防火墙体系主要工作在网络层，针对TCP/IP数据包实施过滤和限制，属于典型的包过滤防火墙（或称网络层防火墙）。Linux的防火墙体系基于内核编码实现，因此具有非常稳定的性能和高效率，也因此被更加广泛采纳和应用。  
**●>Linux防火墙发展历史**  
linux内核从很早的时候就实现了网络防火墙的功能，并为用户提供了管理防火墙规则的命令工具。  
>>>在2.0版本的内核中，包过滤机制是ipfw，管理防火墙规则的命令工具是ipfwadm。  
>>>在2.2版本的内核中，包过滤机制是ipchain，管理防火墙规则的命令工具是ipchains。  
>>>在2.4版本的内核开始，包过滤机制是netfiter，管理防火墙的命令工具是iptables。  
本文以iptables作为主要命令工具，学习Linux防火墙的配置和使用。

**●>netfiter与iptables**

在许多关于防火墙的资料中，netfiter和iptables通常都可以用来指Linux防火墙，往往容易引起混淆。两者的主要区别如下:  
>>>netfiter：指的是Linux内核中实现包过滤防火墙的内部结构，不以程序或文件的形式存在，属于“内核态”（Kernel Space，又称内核空间）的防火墙功能体系  
>>>iptables: 指的是管理防火墙的命令工具，程序通常位于/sbin/iptables，属于“用户态”（User Space，又称用户空间）的防火墙管理体系。  
1.2>、iptables的规则表、链结构

iptables的作用在于为包过滤的实现提供规则（或称策略），通过各种不同的规则，告诉netfiter对来自有些源、前往某些目的地或具有某些协议特征的数据包应该如何处理。为了更方便地组织和管理防火墙策略，iptables采用了“表”和“链”的分层结构。每个规则“表”相当于内核空间的一个容器，根据处理数据包的时机不同，表容器内包括不同的规则“链”。针对特定数据包的各种防火墙规则，按顺序依次放入对应的规则“链”中。

[](http://img1.51cto.com/attachment/201306/150415661.jpg)

◆规则表

按照防火墙策略的不同用途，iptables管理着四个不同的规则表，其功能分别由独立的内核模块实现。

>>>filter表，包含三个规则链：INPUT,FORWARD,OUTPUT。

filter 表主要用于对数据包进行过滤，根据具体的规则决定是否放行该数据包。

filter 表对应的内核模块为 iptable\_filter。

>>>nat表，包含三个规则链：PREROUTING，POSTROUTING，OUTPUT。

nat (Network Address Translation，网络地址转换)表主要用于修改数据包的IP地址、端口号等信息。

nat 表对应的内核模块为 iptable\_nat。

>>>mangle表，包含五个规则链：PREROUTING，POSTROUTING，INPUT，OUTPUT，FORWARD。

mangle表主要用于修改数据包的TOS（Type Of Service，服务类型）、TTL（Time To Live，生存周期）指以及为数据包设置Mark标记，以实现Qos(Quality Of Service，服务质量)调整以及策略路由等应用，由于需要相应的路由设备支持，因此应用并不广泛。

mangle 表对应的内核模块为 iptable\_mangle。

>>>raw表，包含两条规则链：OUTPUT、PREROUTING。

raw表是自1.2.9以后版本的iptables新增的表，主要用于决定数据包是否被状态跟踪机制处理。在匹配数据包时，raw表的规则要优先于其他表。

raw表对应的内核模块为iptable\_raw。

Ps:在iptables的四个规则表中，mangle表和raw的表应用相对较少，下面主要介绍filter表和nat表的防火墙应用。

◆规则链

在处理各种数据包时，根据防火墙规则的不同介入时机，iptables供涉及5种默认规则链，其应用时间点分别对应如下。

>>>INPUT链：当接收到防火墙本机地址的数据包（入站）时，应用此链中的规则。

>>>OUTPUT链：当防火墙本机向外发送数据包（出战）时，应用次链中的规则。

>>>FORWARD链：当接收到需要通过防火墙发送给其他地址的数据包（转发）时，应用次链中的规则。

>>>PREROUTING链：在对数据包作路由选择之前，应用次链中的规则。

>>>POSTROUTING链：在对数据包作路由选择之后，应用次链中的规则。

Ps:期中INPUT、OUTPUT链更多的应用在“主机防火墙”中，即主要针对服务器本机进出数据的安全控制；而FORWARD、PREROUTING、POSTROUTING链更多的应用在“网络防火墙”中，特别是防火墙服务器作为网关使用时的情况。

1.3>、数据包过滤匹配流程

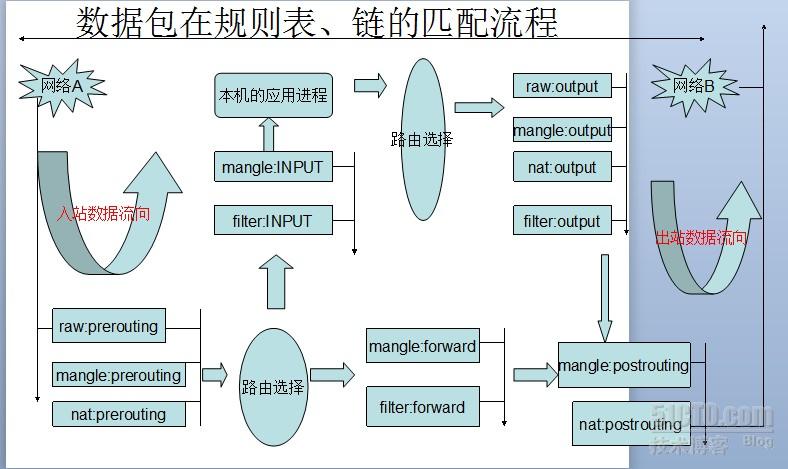
在前面内容中，已经介绍过iptables管理的4个默认表和5种规则链，在每个表的规则链里都可以设置防火墙规则，防火墙将根据这些规则来决定如何处理数据包。那么，当网络数据包到达防火墙以后，会有限使用哪一个表、哪一个链里的规则呢？数据包出入防火墙的规则匹配过程是怎样的？下面从不同的角度依次介绍数据包过滤的匹配流程。

ⅰ>、规则表之间的优先顺序

当数据包抵达防火墙时，将依次应用raw、mangle、nat和filter表中对应链内的规则（如果有的话）就像上图；

ⅱ>、规则链之间的优先顺序

由于默认规则链是根据规则介入时机进行分类的，因此优先顺序直接取决于数据包的具体流程；

[](http://img1.51cto.com/attachment/201306/161312326.jpg)

[ⅰ>、入站数据流向： 来自外界的数据包到达防火墙后，首先被PREROUTING规则链处理（是否修改数据包地址等），之后会进行路由选择（判断该数据包应该发往何处），如果数据包的目标地址是防火墙本机（如Internet用户访问防火墙主机中的Web服务的数据包），那么内核将其传递给INPUT链进行处理（决定是否允许通过等），通过以后再交给系统上层的应用程序（如httpd服务器）进行响应。](http://img1.51cto.com/attachment/201306/161312326.jpg)

ⅱ>、转发数据流向：来自外界的数据包到达防火墙后，首先被PREROUTING规则处理，之后会进行路由选择，如果数据包的目标地址是其他外部地址（如局域网用户通过网关访问QQ站点的数据包），则内核将其传递给FORWARD链进行处理（是否转发或拦截），然后再讲给POSTROUTING规则链（是否修改数据包的地址等）进行处理。

ⅲ>、出战数据流向：防火墙本机向外部地址发送的数据包（如在防火墙主机中测试公网DNS服务时），首先被OUTPUT规则链处理，之后进行路由选择，然后传递给POSTROUTING规则链（是否修改数据包的地址等）进行处理。

1.4>、规则链内部各条防火墙规则之间的优先顺序

在数据包经由各条规则链的处理过程中，依次按第1条规则、第2条规则、第3条规则.....的顺序进行匹配和处理。如果找到一条能够匹配该数据包的规则，则不会继续检查后面的规则（使用LOG记录日志的规则除外，下面iptables配置说明）。如果比对完整个规则链，也找不到和数据包相匹配的规则，就按照该规则链的默认策略进行处理。

2、管理和设置iptables规则

前面讲解了linux防火墙的表、链结构，下面讲述netfilter防火墙的管理工具——iptables命令的使用。

2.1 >、iptables的基本语法格式

使用iptables命令设置防火墙规则时，其基本的命令格式如下：

iptables [ -t 表名] 命令选项 [链名] [条件匹配] [-j 目标动作或跳转]

其中，表名、链名用于指定iptables命令所操作的表和链，若未指定表名，将默认使用filter表；命令选项用于指定管理iptables规则的方式，如插入、增加、删除以及查看等；条件匹配用于指定对符合什么样条件的数据包进行处理；目标动作或跳转用于知道数据包的处理方式，如允许通过、拒绝、丢弃或跳转（jump）给其他链进行处理等。

2.2>、管理iptables规则

在管理iptables防火墙的过程中，首先需要考虑的就是如何查看规则、添加规则、清空链内的规则等基本操作。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | iptables 命令的管理控制选项  选项名     功能及特点  -A      在指定链的末尾添加（--append）一条新的规则  -D      删除（--delete）指定链中的某一条规则，按规则序号或内容确定要删除的规则  -I      在指定链中插入（--insert）一条新的规则，若未指定插入位置，则默认在链的开头插入  -R      修改、替换（--replace）指定链中的某一条规则，按规则序号或内容确定要替换的规则  -L      列出（--list）指定链中的所有的规则进行查看，若未指定链名，则列出表中所有链的内容  -F      清空（--flush）指定链中的所有规则，若未指定链名，则清空表中所有链的内容  -N      新建（--new-chain）一条用户自己定义的规则链  -X      删除指定表中用户自定义的规则链（--delete-chain）  -P      设置指定链的默认策略（--policy）  -n      使用数字形式（--numeric）显示输出结果，若显示主机的IP地址而不是主机名  -v      查看规则列表时显示详细（--verbose）的信息  -V      查看iptables命令工具的版本（--Version）信息  -h      查看命令帮助信息（--help）  --line-number   查看规则列表时，同时显示规则在链中的顺序号 |

其中，添加、插入、删除、清空规则和查看规则是最常用的管理选项，下面通过实例的方式显示部分选项的应用。

Α、添加及插入规则

eg     :在filter表的INPUT链的末尾添加一条防火墙规则。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -t filter -A INPUT -p tcp -j ACCEPT |

eg :  在filter表的INPUT链中插入一条防火墙规则（此处省略了“-t filter”选项，按默认处理filter表）。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -I INPUT -p udp -j ACCEPT |

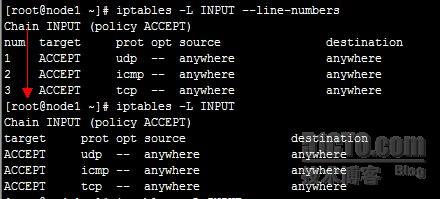
eg:   在filter表的INPUT链中插入一条防火墙规则（作为链中的第二条规则）。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -I INPUT 2 -p icmp -j ACCEPT |

B、查看规则列表

eg:   查看filter表INPUT链中的所有规则，同时显示各条规则的顺序号。（下面是对比图）

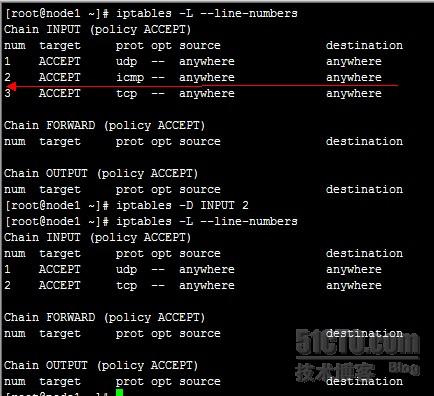
|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -L INPUT --line-numbers |
| 1 | [root@node1 ~]# iptables -L INPUT |

[](http://img1.51cto.com/attachment/201306/170056403.jpg)

C、删除、清空规则

eg:   删除filter表INPUT链中的第2条规则。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -D INPUT 2 |

[](http://img1.51cto.com/attachment/201306/170737918.jpg)

eg： 清空filter表、nat表、mangle表各链中的所有规则。

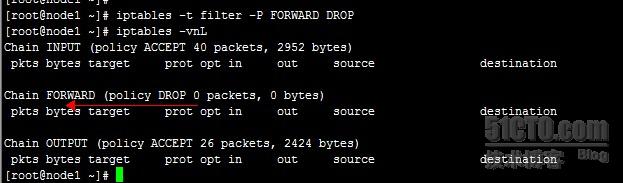
|  |  |
| --- | --- |
| 1  2  3 | [root@node1 ~]# iptables -F             //不指定表名时默认清空filter表  [root@node1 ~]# iptables -t nat -F  [root@node1 ~]# iptables -t mangle -F |

D、设置规则链的默认策略

最基本的两种策略为ACCEPT（允许）、DROP（丢弃）。

eg:  将filter表中FORWARD规则链的默认策略设为DROP

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -t filter -P FORWARD DROP |

[](http://img1.51cto.com/attachment/201306/171251760.jpg)

E、获得iptables相关选项用法的帮助信息

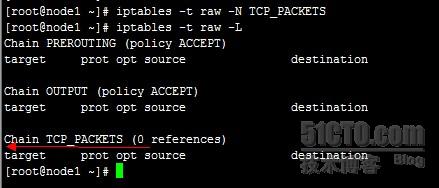
eg:  查看iptables命令中关于icmp协议的帮助信息（在输出内容的最后部分列出）。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -p icmp -h |

F、新增、删除自定义规则链

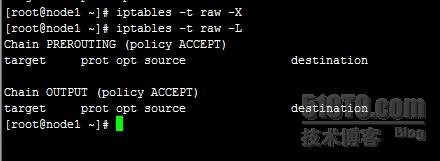
eg:  在raw表中新增一条自定义的规则链，链名为TCP\_PACKETS.

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -t raw -N TCP\_PACKETS |

[](http://img1.51cto.com/attachment/201306/171721658.jpg)

eg:  清空raw表中的用户自定义的所有规则链。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -t raw -X |

[](http://img1.51cto.com/attachment/201306/171901449.jpg)

Ps：在iptables的规则表中，允许用户自定义新的链。但是需要在默认的链中添加相应的跳转策略（如“iptables -I INPUT ...... -j 自定义链名”），否则在处理数据包时将不会使用自定义链中的规则。后面会给个例子[http://img.baidu.com/hi/jx2/j_0009.gif](http://img.baidu.com/hi/jx2/j_0009.gif)；好累~~~

2.3>、条件匹配

对于Linux防火墙来说，应该对什么样的数据包进行过滤，对什么样的数据包做地址转换......，对于一条有效的防火墙策略来说，条件匹配的设置起着决定性的作用，Linux防火墙需要非常清楚的知道针对符合什么条件的数据包进行什么样的处理。

区分数据包的条件匹配方式可以分为通用匹配、隐含匹配和显式匹配，各种匹配条件可以结合在一起使用。

A、通过（general）条件匹配

这种匹配操作一般可以直接使用，而不依赖于其他的条件匹配及其扩展。常见的通用匹配方式包括以下集中。

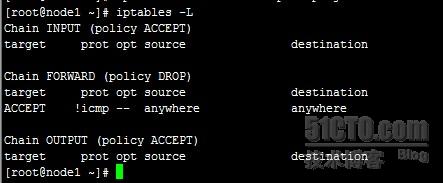
>>>协议匹配。用于检查数据包的网络协议（--protocal），允许使用的协议名包含在/etc/protocols文件中，常见的为tcp、udp、icmp和all（针对所有IP数据包）。在iptables命令中使用“-p 协议名”的形式。

eg:  拒绝进入防火墙的所有icmp协议数据包。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -I INPUT -p icmp -j REJECT |

eg  :允许防火墙转发除icmp协议以外的所有数据包（使用惊叹号"!"可以将除条件取反）。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -A FORWARD ! -p icmp -j ACCEPT |

[](http://img1.51cto.com/attachment/201306/203022288.jpg)

>>>地址匹配。用于检查数据包的IP地址、网络地址。在iptables命令中使用“-s 源地址”和“-d 目标地址”的形式，分别对应于源（--source）地址和目标（--destination）地址。

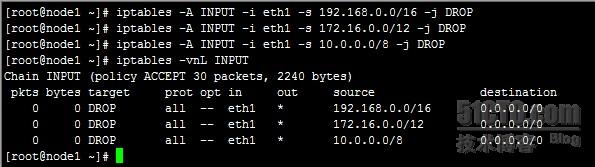
eg:  拒绝转发来自192.168.1.11主机的数据，允许转发来自192.168.0.0/24网段的数据。

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]# iptables -A FORWARD -s 192.168.1.11 -j REJECT  [root@node1 ~]# iptables -A FORWARD -s 192.168.0.0/24 -j ACCEPT |

>>>网络接口匹配。用于检查数据包从防火墙的哪一个接口进入或离开。在iptables命令中使用“-i 网络接口名”和“-o 网络接口名”的形式，分别对应于接收数据包（--in-interface）的网卡和发送数据包（--out-interface）的网卡

eg: 丢弃从外网接口（eth1）进入防火墙本机的源地址为私网地址的数据包。

|  |  |
| --- | --- |
| 1  2  3 | [root@node1 ~]# iptables -A INPUT -i eth1 -s 192.168.0.0/16 -j DROP  [root@node1 ~]# iptables -A INPUT -i eth1 -s 172.16.0.0/12 -j DROP  [root@node1 ~]# iptables -A INPUT -i eth1 -s 10.0.0.0/8 -j DROP |

[](http://img1.51cto.com/attachment/201306/204243720.jpg)

eg: 管理员在网关服务器上检测到来自某个IP网段（如 10.20.30.0/24）的频繁扫描，希望设置iptables规则封堵该IP地址段，两小时后解封。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [root@node1 ~]# iptables -I INPUT -s 10.20.30.0/24 -j DROP  [root@node1 ~]# iptables -I FORWARD -s 10.20.30.0/24 -j DROP  [root@node1 ~]# at now +2 hours  at> iptables -D INPUT 1  at> iptables -D FORWARD 1  at><EOT>                             //此处按Ctrl+D组合键  job 4 at 2013-06-11 22:47 |

B、隐含（Implicit）条件匹配

这种匹配通常需要以指定的协议匹配为前提，其对应的功能由iptables自动（隐含）的装载入内核，一般不需要用户收到加载模块。常见的隐含匹配方式包括以下几种。

>>>端口匹配。用于检查数据包的TCP或UDP端口号，需要以“-p tcp”或“-p udp”匹配为前提。在iptables命令中使用“--sport 源端口号”、“--dport 目标端口”的形式，分别对应于源端口（--source-port）、目标端口（--destination-port）。端口可以表示为单个端口号或者用冒号“：”分割的端口范围。

eg：  仅允许系统管理员从202.13.0.0/16 网段使用SSH方式远程登录防火墙主机。

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]# iptables -A INPUT -p tcp --dport 22 -s 202.13.0.0/16 -j ACCEPT  [root@node1 ~]# iptables -A INPUT -p tcp --dport 22 -j DROP |

eg:  允许本机开放从TCP端口20-1024提供的应用服务

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]# iptables -A INPUT -p tcp --dport 20:1024 -j ACCEPT  [root@node1 ~]# iptables -A OUTPUT -p tcp --sport 20:1024 -j ACCEPT |

eg:  作为网关使用时，允许转发来自192.168.0.0/24局域网段的DNS解析请求数据包。

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]# iptables -A FORWARD -s 192.168.0.0/24 -p udp --dport 53 -j ACCEPT  [root@node1 ~]# iptables -A FORWARD -d 192.168.0.0/24 -p udp --sport 53 -j ACCEPT |

>>>、TCP标记匹配。用于检查数据包的TCP标记位（--tcp-flags），需要以“-p tcp”匹配为前提。在iptables命令中使用“--tcp-flags检查范围被设置的标记”的形式，两个参数“检查范围”和“被设置的标记”均为TCP标记的列表，各标记之间用逗号分割。“检查范围”告诉iptables需要检查数据包的哪几个标记，“被设置的标记”则明确匹配对应值为1的标记。

eg:  拒绝从接口（eth0）直接访问防火墙本机的数据包，但是允许响应防火墙TCP请求的数据包进入。

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]# iptables -P INPUT DROP     //此命令执行小心，不提前设置通过ssh的话，你的远程直接就断开了  [root@node1 ~]# iptables -I INPUT -i eth0 -p tcp --tcp-flags SYN,RST,ACK SYN -j REJECT |

>>>、ICMP类型匹配。用于检查ICMP数据包的类型（--icmp-type）,有选择的过滤数据包，需要以“-p icmp”匹配为前提。在iptables命令中使用“--icmp-type ICMP类型”的形式。ICMP类型可以使用字符串或数字代码，例如“Echo-Request”(数字代码为8)、“Echo-Reply”(数字代码为0)、“Destination-Unreachable”(数字代码为3)，分别对应ICMP协议的请求、回显、目标不可达数据。

eg:  禁止其他主机ping防火墙主机，但是允许从防火墙上ping其他主机（允许接受ICMP回应数据）。

|  |  |
| --- | --- |
| 1  2  3 | iptables -A INPUT -p icmp --icmp-type Echo-Request -j DROP  iptables -A INPUT -p icmp --icmp-type Echo-Reply -j ACCEPT  iptables -A INPUT -p icmp --icmp-type destination-Unreachable -j ACCEPT |

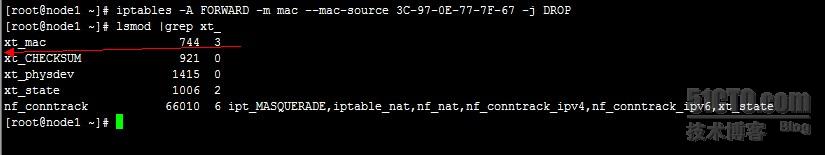
C、  显式（Explicit）条件匹配

这种匹配的功能需要由额外的内核模块提供，因此需要手工指定匹配方式。在iptables命令中使用“-m 模块关键字”的形式调用显式匹配，当然，还的指定对应的匹配内容。在使用过显式匹配以后，可以执行“lsmod |grep xt\_”命令查看到扩展防火墙功能的相关内核模块（如：xt\_mac，xt\_state，xt\_multiport等）。常见的显式匹配包括以下几种。

>>>、MAC地址匹配。主要用于检查数据包的源MAC地址。在iptables命令中使用“--mac-source MAC地址”的形式。

eg:  禁止转发来自MAC地址为3C-97-0E-77-7F-67（这是我本机的mac）的主机的数据包

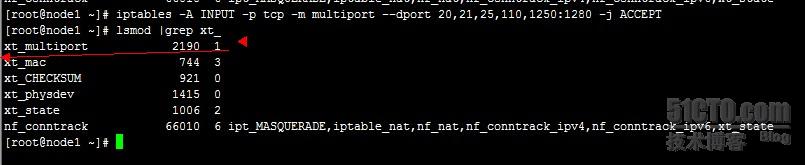
|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -A FORWARD -m mac --mac-source 3C-97-0E-77-7F-67 -j DROP |

[](http://img1.51cto.com/attachment/201306/222715492.jpg)

>>>、多端口匹配。检查数据包的源端口、目标端口时，用于匹配多个不连续的端口号。在iptables命令中主要使用”--dports 端口列表“或者”--sports 端口列表“的形式，分别对应源端口地址列表、目标端口地址列表。

eg :  允许防火墙本机对外开放TCP端口20、21、25、110以及被动模式FTP端口1250-1280。

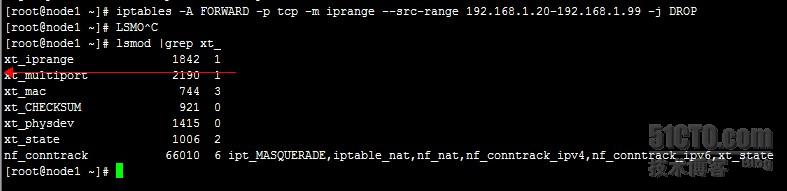
|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -A INPUT -p tcp -m multiport --dport 20,21,25,110,1250:1280 -j ACCEPT |

[](http://img1.51cto.com/attachment/201306/223516717.jpg)

>>>、多IP地址匹配。检查数据包的源地址、目标地址时，用于匹配一段范围内的IP地址。在iptables命令中主要使用”--src-range IP地址范围“或者”--dst-range IP地址范围“的形式，分别对应源IP地址范围、目标ip地址范围。

eg： 禁止转发源IP地址为192.168.1.20~192.168.1.99 的TCP数据包。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -A FORWARD -p tcp -m iprange --src-range 192.168.1.20-192.168.1.99 -j DROP |

[](http://img1.51cto.com/attachment/201306/224040884.jpg)

>>>、状态匹配。基于iptables的状态跟踪机制，检查数据包的链接状态（State）。常见的数据包状况主要包括NEW(与任何连接无关的)、ESTABLISHED(响应请求或者已建立连接的)和RELATED(与已有连接有相关性的，如FTP数据连接)。

eg:  禁止转发与正常TCP连接无关的非--syn请求数据包（如网络中可能存在的一些非法攻击数据包）。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -A FORWARD -m state --state NEW -p tcp ! --syn -j DROP |

eg : 拒绝访问防火墙的新数据包，但允许响应连接或与已有连接相关的数据包。

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]# iptables -A INPUT -p tcp -m state --state NEW -j DROP  [root@node1 ~]# iptables -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT |

eg:  在服务器上设置防火墙策略，只开放本机的Web服务（80端口）、FTP服务（21、20端口、20450~20480），放行外部主机发往服务器其他端口的应答数据包，将其他入站数据包均予以丢弃处理。

|  |  |
| --- | --- |
| 1  2  3  4 | [root@node1 ~]# iptables -I INPUT -p tcp -m multiport --dport 20,21,80 -j ACCEPT  [root@node1 ~]# iptables -I INPUT -p tcp --dport 20450:20480 -j ACCEPT  [root@node1 ~]# iptables -I INPUT -p tcp -m state --state ESTABLISHED -j ACCEPT  [root@node1 ~]# iptables -P INPUT DROP |

2.4>、数据包控制

在上面通过一些iptables规则的实例，已经接触到了如ACCEPT、DROP、REJECT等数据包控制方式。iptables对满足”条件匹配“指定条件的数据包，最常采用的处理方式如下。

>>>>>、ACCEPT    允许数据包通过。

>>>>>、DROP       直接丢弃数据包，不给出任何回应信息。

>>>>>、REJECT     拒绝数据包通过，必须时会给数据发送端一个响应信息。

>>>>>、LOG         在/var/log/messages 文件中记录日志信息，然后将数据包传递给下一条规则。通过情况下，只要在规则链中找到一条相匹配的规则，则不再继续检查该链内后面的规则；但使用LOG处理方式的规则是一个特例，因为LOG只是一个辅助动作，并没有真正的处理数据包。

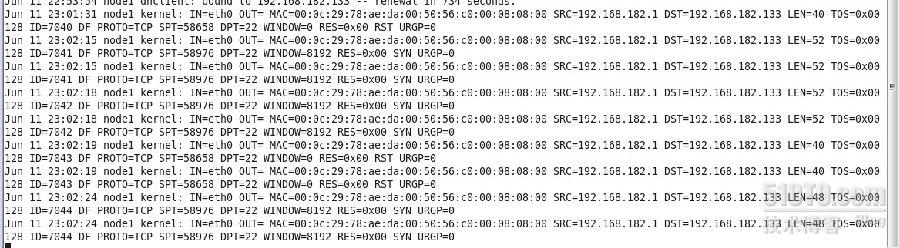
eg：  对于尝试通过SSH方式登录防火墙主机的访问数据，记录日志信息并禁止其访问。

|  |  |
| --- | --- |
| 1  2 | [root@node1 桌面]# iptables -I INPUT -p tcp --dport 22 -j DROP  [root@node1 桌面]# iptables -I INPUT -p tcp --dport 22 -j LOG |

这个时候当我在ssh的时候，会发现如下截图的情况：

[](http://img1.51cto.com/attachment/201306/230327104.jpg)

tail -f /var/log/messages 查看本机log发现

[](http://img1.51cto.com/attachment/201306/230412721.jpg)

为了避免日志记录过于频繁，通过结合LIMIT显式匹配（-m limit）对日志写入频繁进行限制。例如以下规则，用于将记录日志的频率限制为平均三次/分钟，允许的峰值为八次。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -R INPUT 1 -p tcp --dport 22 -m limit --limit 3/minute --limit-burst 8-j LOG |

>>>、 用户自定义链 将数据包传递给用户自定义的链（内的规则）进行处理。

eg：自定义一个新的链MY1，转发自/至192.168.1.0/24网段的数据包均交给该链中的规则处理。

|  |  |
| --- | --- |
| 1  2  3  4  5 | [root@node1 ~]# iptables -t filter -N MY1  [root@node1 ~]# iptables -A FORWARD -s 192.168.1.0/24 -j MY1  [root@node1 ~]# iptables -A FORWARD -d 192.168.1.0/24 -j MY1  [root@node1 ~]# iptables -A MY1 -p icmp -j DROP  [root@node1 ~]# .....接着可以添加更多规则 |

# linux之iptables详解及配置（二）+案例

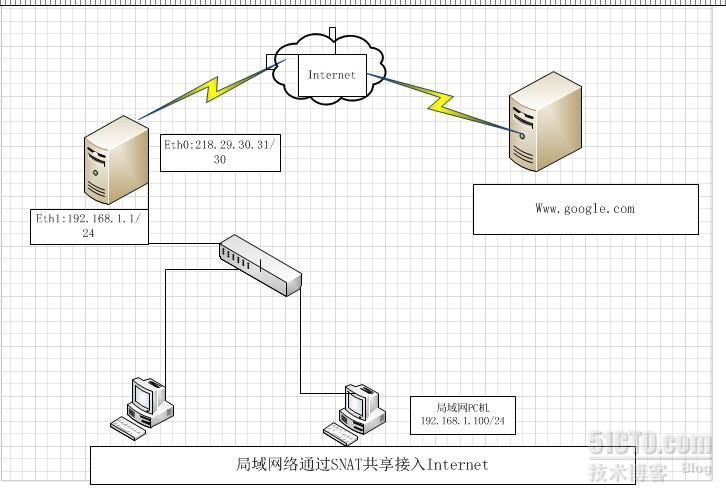
1、SNAT策略及应用

>>>SNAT策略概述

  随着Internet网络在全世界范围内的快速发展，IPv4协议支持的可用IP地址资源逐渐变得山穷水尽，资源匮乏使得很多企业难以申请更多的公网IP地址，或者只能承受一个或者少数几个公网IP地址的费用。而与此同时，大部分企业都面临着将局域网的主机接入Internet的需求。使用iptables的SNAT策略，可以基本化解这个难题。

案例：1、

在一个小型的企业网络中，Linux网关服务器有两块网卡，分别用于连接内部局域网和Internet；如下图

[](http://img1.51cto.com/attachment/201306/145744535.jpg)

当Linux网关服务器正常开启路由转发（未使用SNAT）时，局域网访问Internet的数据包，经网关转发后其源IP地址保持不变（仍为192.168.1.100），而Internet中主机收到这样的请求数据包后，将无法为其返回应答数据包（因为该地址是私有IP地址，无法为其正确路由），从而导致访问失败；

如果在Linux网关服务器中有针对性的应用SNAT策略，数据包转发情况就不一样了。当局域网访问Internet的数据包到达网关服务器时，网关进行路由选择，若发现该数据包需要从外网接口（如eth0）向外转发，则将该数据包的源IP地址（如：192.168.1.100）修改为网关的外网接口地址（如：218.29.30.31），然后才提交到Internet中，最后发送到目标主机（www.google.com）。相当于以网关服务器的公网身份提交数据访问请求，自然就可以收到正常的响应数据包。然后网关服务器再根据之前建立的SNAT映射，将响应数据包返回给局域网中的源主机。只要连接到第一个包被SNAT处理了，那么这个连接及对应数据流的其他包通常也会自动的被进行SNAT处理。

>>>SNAT策略应用；（解决上述问题）

从前面的需求中，大致了解到，SNAT典型应用于局域网共享上网的接入，而处理数据包的切入时机，主要选择在路由选择之后（POSTROUTING）进行。SNAT的关键在于将局域网外发数据包的源IP地址（私有地址）修改为网关的外网接口IP地址（公有地址）。

SNAT只能用于nat表的POSTROUTING链。使用iptables命令设置SNAT策略时，需要结合“--to-source IP地址”选项指定修改后的IP地址（例如：-j SNAT --to-source 218.29.30.31）.

下面通过实例配置来说明SNAT策略的用法，案例环境及需求描述如下。（拓扑图如上）

》、公司的网关服务器使用了Linux操作系统。

》》、网关上有两块网卡： 期中eth0连接Internet，使用固定IP地址 218.29.30.31/30；eth1连接局域网，使用固定IP地址192.168.1.1/24。

》》》、局域网内各主机的默认网关设置为192.168.1.1，且已经设置了正确的DNS服务器。

》》》》、现需要在Linux网关主机中进行正确配置，以使192.168.1.0/24网段的局域网用户能够共享方式访问Internet。

根据上述环境，在Linux网关中推荐的操作步骤如下。

(1)、确认开启网关服务器的路由转发功能。

|  |  |
| --- | --- |
| 1  2  3 | [root@node1 ~]# vim /etc/sysctl.conf  net.ipv4.ip\_forward = 1  [root@node1 ~]# sysctl -p |

(2)、为局域网访问Internet的数据包采用SNAT策略，将源地址更改为服务器的公网IP地址。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j SNAT --to-source218.29.30.31 |

使用局域网内的客户机访问Internet中的网站（如：www.baidu.com）进行测试，测试成功以后将上述SNAT策略添加到防火墙脚本文件中，并设置未开机后自动运行。

Ps 以上介绍的应用是网关服务器具有固定的静态公网IP地址的情况，而在某些时候，企业很可能使用非固定的动态公网地址，例如使用ADSL宽带接入时可能获得是动态IP地址。那么在这种网络环境下，又应该如何设置SNAT策略呢？iptables提供了一个名为MASQUERADE(伪装)的数据包处理方式，MASQUERADE相当于SNAT的特例，同样也完成修改（伪装）数据包源IP地址的工作，只不过它会自动获取外网接口的IP地址，而无需使用“--to-source”的形式。

  使用iptables命令设置MASQUERADE策略时，只需要去掉SNAT策略中的“--to-source IP地址”，改用“-j MASQUERADE”指定数据包处理方式即可。ADSL通常使用PPPOE技术，在Linux系统中，对应的连接名称为ppp0、ppp1等，如果无法确定连接的编号，还可以使用ppp+代替。

eg：  设置MASQUERADE策略，使用192.168.1.0/24网段能够通过网关的ppp0连接共享上网。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ppp0 -j MASQUERADE |

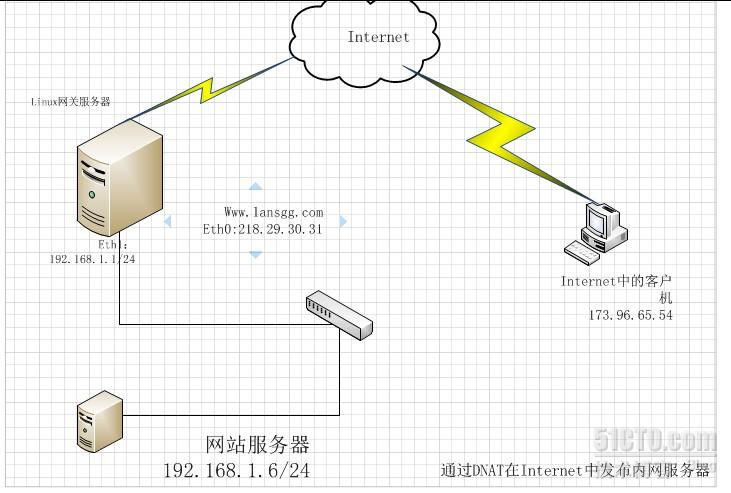
当然，即使网关服务器有静态的IP地址，也同样可以使用MASQUERADE，而不用SNAT。只不过，MASQUERADE会比SNAT多一些额外的开销，因此如果有固定的IP地址，最好还是使用SNAT策略。

2、DNAT策略及应用

>>>DNAT策略概述

DNAT策略与SNAT非常相似，只不过应用方向反过来了。SNAT用于修改源IP地址，而DNAT用于修改目标IP地址；SNAT只能在nat表的POSTROUTING链中，而DNAT只能用在nat表的PREROUTING链和OUTPUT链（或被其调用的链）中。

 例如：考虑到应用服务的安全和稳定性，公司将对外的Web服务器架设在一个内部网络中，而公司仍然只有一个公网IP地址，又需要使Internet中的客户机能够访问公司的网站，这时候，使用DNAT策略可以有效保证位于内网中的网站服务器于Internet中的客户机的相互通讯。

[](http://img1.51cto.com/attachment/201306/160630175.jpg)

由于企业申请的Internet域名要解析到一个合法的公网IP地址（如：218.29.30.31），当Internet中的客户机提交的http请求到达企业的网关主机时，网关首先判断该数据包的目标地址和目标端口，若发现该数据包需要访问本机的80端口，则将该数据包的目标IP地址（如：218.29.30.31）修改为内网中真正的网站服务器的IP地址（如：192.168.1.6），然后才进行路由选择，最后发送到内部的网站服务器（192.168.1.6）.然后网关服务器再根据之前建立的DNAT映射，修改返回的HTTP应答数据包的源IP地址，最后返回给Internet中的客户机。

在上述的DNAT转换地址的过程，Internet中的客户机无需指定企业网站服务器的真实局域网地址，中间的转换完全由网关主机完成。通过DNAT策略，位于企业内部的应用服务器就可以对Internet提供服务了。

>>>DNAT策略的应用

   使用iptables命令设置DNAT策略时，需要结合"--to-destination IP地址"的选项指定企业内网服务器的实际IP地址（如：-j DNAT --to-destination 218.29.30.31）。下面将通过两个实例来说明DNAT策略的用法。

案例：1、见上图拓扑：

》、公司在ISP注册了域名 www.lansgg.com，并对应于Linux网关的外网的接口（eth0）地址：218.20.30.31.

》》、公司的网站服务器位于局域网内，IP地址为192.168.1.6。

》》》、Internet用户可以通过访问[www.lansgg.com来查看公司的网站内容。](http://www.lansgg.xn--com-ww1ene96nbxr4ww66ar98gmhbb9x5so./)

根据上述环境，在Linux网关中推荐的操作步骤如下：

（1）确认开启网关服务器的路由转发功能。

|  |  |
| --- | --- |
| 1  2  3 | [root@node1 ~]# vim /etc/sysctl.conf  net.ipv4.ip\_forward = 1  [root@node1 ~]# sysctl -p |

（2）在网关上添加DNAT映射，对于访问网关80端口的数据包，将目标地址更改为网站服务器的内网IP地址。

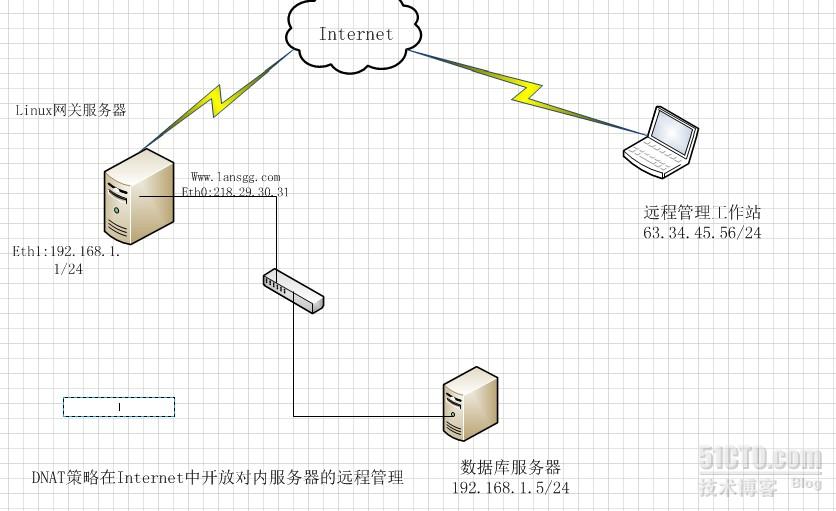
|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -t nat -A PREROUTING -i eth0 -d 218.29.30.31 -p tcp --dport 80 -j DNAT --to-destination 192.168.1.6 |

使用Internet中客户机访问http://www.lansgg.com进行测试，测试成功以后将上述DNAT策略添加到网关的防火墙脚本文件中，并设置为开机后自动运行。

Ps: 如果没有设置SNAT共享上网策略，内网的客户机可能将无法使用公网地址（如：218.29.30.31）的形式访问公司内部的网站服务器。若确实有这种需求，可以有针对性的添加一条SNAT规则，如下命令：

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -t nat -I POSTROUTING -s 192.168.1.0/24 -d 218.29.30.31 -p tcp --dport80 -j SNAT --to-source 192.168.1.1 |

案例：2、若需要做DNAT的内部服务与网关主机本身的服务发送冲突，则需要对数据访问进行区分。例如，Linux网关本身运行了SSH服务，内部网络中的数据库服务器也运行了SSH服务，两台主机都希望能从Internet中进行远程管理。由于可用的公网IP地址只有一个，因此就有必要在网关上根据访问端口进行区分。SSH服务器默认使用的端口号为22。

[](http://img1.51cto.com/attachment/201306/164403693.jpg)

》、公司的Linux网关服务器的外网IP地址eth0:218.29.30.31；位于局域网的数据库服务器IP地址：192.168.1.5；

》》、根据公司安排，管理员要能够随时从家里的主机远程管理数据库服务器，登录的目标地址为218.29.30.31：2222

》》》、管理员家里主机的IP地址可能在63.34.45.0/24网段范围中随机变动。

根据上述环境，应用DNAT策略的思路如下。

>、访问218.29.30.31:22的数据仍保持默认（不做处理），对应网关主机本身的SSH服务。

>>、两台主机的SSH服务都使用默认的22端口不变。

>>>、在网关主机上，将访问本机2222端口的数据包进行DNAT处理，对应到内部网络中的数据库服务器（需要同时指定目标端口）。

推荐操作步骤如下；

（1）、在数据库服务器上开启ssh服务，（注意设置好路由或默认网关记录）

|  |  |
| --- | --- |
| 1  2 | service sshd start  route add -net 63.34.45.0/24 gw 192.168.1.1 |

（2）、确认开启网关服务器的路由转发功能。

|  |  |
| --- | --- |
| 1  2  3 | [root@node1 ~]# vim /etc/sysctl.conf  net.ipv4.ip\_forward = 1  [root@node1 ~]# sysctl -p |

（3）、在网关上添加DNAT映射，对于访问网关2222端口的数据包，将目标地址更改为数据库服务器主机的内网IP地址

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables -t nat -A PREROUTING -i eth0 -s 63.34.45.0/24 -d 218.29.30.31 -p tcp --dport 2222 -j DNAT --to-destination 192.168.1.5:22 |

经测试成功以后，将上述DNAT策略添加到防火墙脚本文件中，并设置为开机后自动运行。

下面是防火墙脚本方面的东西；

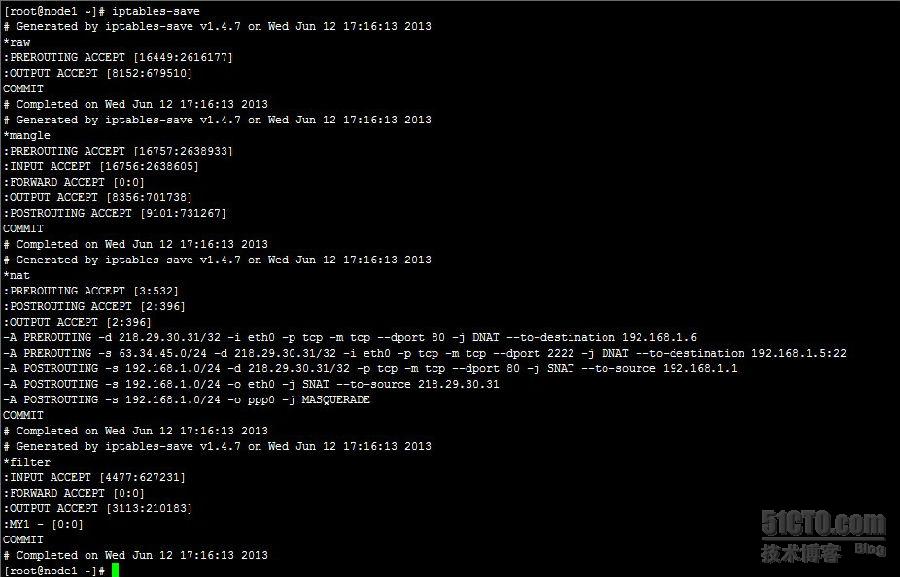
前面已经学会了iptables命令的语法和简单的规则设置。在添加防火墙规则时，必须充分理解数据包匹配的流程，尤其是规则链内的匹配流程。这样才能更好的优化规则顺序，提高包过滤机制的效率和准确性。

3、导出、导入防火墙规则

iptables软件包提供了两个命令:iptables-save、iptables-restore，分别用于保存（save）和恢复（restore）防火墙规则，使用这两个命令可以很方便的导出和导入规则。

>>>>>、iptables-save

直接执行iptables-save命令时，将会把当前设置的防火墙规则信息输出到终端。

[](http://img1.51cto.com/attachment/201306/171638500.jpg)

通常情况下，可以使用重定向将信息保存为指定的配置文件，结合系统默认提供的iptables服务脚本，可以自动加载位于/etc/sysconfig/iptables文件中的规则设置。

eg：将当前调试好的iptables规则保存到配置文件，并通过iptables服务脚本自动加载。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [root@node1 ~]# iptables-save >/etc/sysconfig/iptables  [root@node1 ~]# service iptables restart  iptables：清除防火墙规则：                                 [确定]  iptables：将链设置为政策 ACCEPT：raw mangle nat filter     [确定]  iptables：正在卸载模块：                                   [确定]  iptables：应用防火墙规则：                                 [确定]  [root@node1 ~]# chkconfig --level 35 iptables on |

>>>、iptables-restore

使用iptables-restore命令，可以从已保存的配置文件中导入iptables规则（该文件必须是使用iptables-save命令导出的配置数据）。

eg : 从已保存的规则配置文件中导入iptables规则。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# iptables-restore </etc/sysconfig/iptables |

# iptables之mangle表应用实现策略路由+（案例）

iptables之mangle表应用实现策略路由；

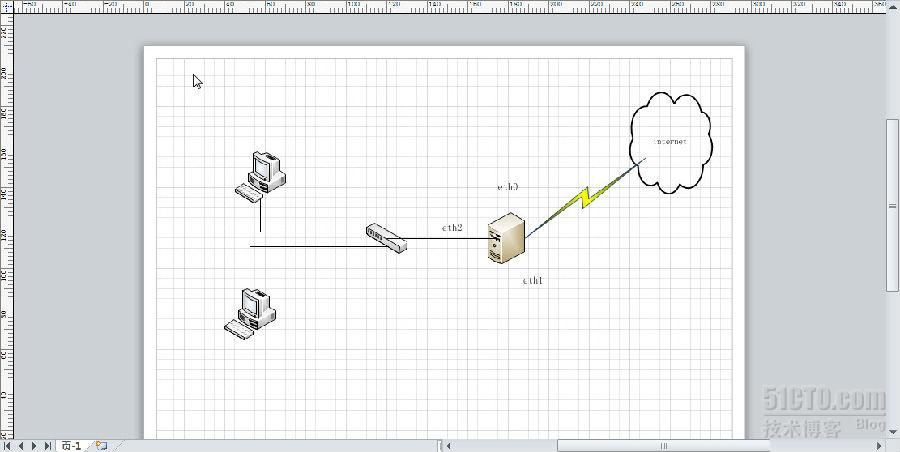
前面的文章已经讲解了：mangle表主要用于修改数据包的TOS（Type Of Service，服务类型）、TTL（Time To Live，生存周期）指以及为数据包设置Mark标记，以实现Qos(Quality Of Service，服务质量)调整以及策略路由等应用，由于需要相应的路由设备支持，因此应用并不广泛。

mangle 表对应的内核模块为 iptable\_mangle。

修改IP包头的TTL值伪装系统版本；

修改IP包头的DSCP值或对特定的数据包设置QOS；不同业务的传送优先级、带宽使用率都可以设置。

下面就来案例说明一下实用且常用的用途吧：

[](http://img1.51cto.com/attachment/201306/211230711.jpg)

案例1、

iptables网关服务器三块网卡：eth0(网通ip：10.0.0.1)、eth1（电信ip:20.0.0.1）；eth2：网关192.168.10.1。

要求：公司内网要求192.168.10.1---100以内的ip使用 10.0.0.1 网关上网（网通），其他IP使用 20.0.0.1 （电信）上网；

iptables网关服务器配置如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | ip route add default gw 20.0.0.1  ip route add table 10 via 10.0.0.1 dev eth0  #eth0 是10.0.0.1所在的网卡,10 是路由表的编号  ip rule add fwmark 10  table 10  #fwmark 10 是标记，table 10 是路由表10。 标记了 10 的数据使用table10 路由表  iptables -A PREROUTING -t mangle -i eth2 -s 192.168.10.1 - 192.168.10.100 -j MARK --set-mark 10  #使用iptables给相应的数据打上标记 |

mangle应用顺序要高于nat、filter哈。

案例2、

iptables网关服务器三块网卡：eth0(网通ip：10.0.0.1)、eth1（电信ip:20.0.0.1）；eth2：网关192.168.10.1。

要求：公司内网要求员工访问外面的网站服务；使用 10.0.0.1 网关上网（网通），其他IP使用 20.0.0.1 （电信）上网；

|  |  |
| --- | --- |
| 1  2  3  4 | iptables -t mangle -A PREROUTING -i eth2 -p tcp --dport 80 -j MARK --set-mark 20  ip route add default gw 20.0.0.1  ip route add table 20 via 10.0.0.1 dev eth0  ip rule add fwmark 20 table 20 |