# **防火墙**

从逻辑上讲：防火墙可以大体分为主机防火墙和网络防火墙。网络防火墙和主机防火墙并不冲突，可以理解为，网络防火墙主外（集体）， 主机防火墙主内（个人）。

①主机防火墙：针对于单个主机进行防护。

②网络防火墙：往往处于网络入口或边缘，针对于网络入口进行防护，服务于防火墙背后的本地局域网。

从物理上讲：防火墙可以分为硬件防火墙和软件防火墙。

①硬件防火墙：在硬件级别实现部分防火墙功能，另一部分功能基于软件实现，性能高，成本高。

②软件防火墙：应用软件处理逻辑运行于通用硬件平台之上的防火墙，性能低，成本低。

# **[iptables概念](https://www.zsythink.net/archives/1199)**

iptables其实不是真正的防火墙，我们可以把它理解成一个客户端代理，用户通过iptables这个代理，将用户的安全设定执行到对应的”安全框架”中，这个”安全框架”才是真正的防火墙，这个框架的名字叫netfilter，netfilter位于内核空间。iptables其实是一个命令行工具，位于用户空间，我们用这个工具操作真正的框架。

netfilter/iptables（下文中简称为iptables）组成Linux平台下的包过滤防火墙，与大多数的Linux软件一样，这个包过滤防火墙是免费的，它可以代替昂贵的商业防火墙解决方案，完成封包过滤、封包重定向和网络地址转换（NAT）等功能。

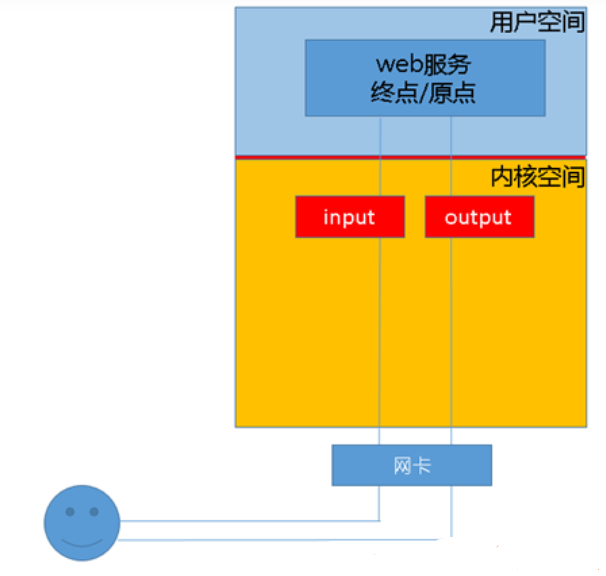
Netfilter是Linux操作系统核心层内部的一个数据包处理模块，它具有如下功能：网络地址转换(Network Address Translate)、数据包内容修改以及数据包过滤的防火墙功能

规则（rules）

规则：根据指定的匹配条件来尝试匹配每个流经此处的报文，一旦匹配成功，则由规则后面指定的处理动作进行处理；

规则其实就是网络管理员预定义的条件，规则一般的定义为”如果数据包头符合这样的条件，就这样处理这个数据包”。规则存储在内核空间的信息包过滤表中，这些规则分别指定了源地址、目的地址、传输协议（如TCP、UDP、ICMP）和服务类型（如HTTP、FTP和SMTP）等。当数据包与规则匹配时，iptables就根据规则所定义的方法来处理这些数据包，如放行（accept）、拒绝（reject）和丢弃（drop）等。配置防火墙的主要工作就是添加、修改和删除这些规则。

当客户端访问服务器的web服务时，客户端发送报文到网卡，而tcp/ip协议栈是属于内核的一部分，所以，客户端的信息会通过内核的TCP协议传输到用户空间中的web服务中，而此时，客户端报文的目标终点为web服务所监听的套接字（IP：Port）上，当web服务需要响应客户端请求时，web服务发出的响应报文的目标终点则为客户端，这个时候，web服务所监听的IP与端口反而变成了原点，我们说过，netfilter才是真正的防火墙，它是内核的一部分，所以，如果我们想要防火墙能够达到”防火”的目的，则需要在内核中设置关卡，所有进出的报文都要通过这些关卡，经过检查后，符合放行条件的才能放行，符合阻拦条件的则需要被阻止，于是，就出现了input关卡和output关卡，而这些关卡在iptables中不被称为”关卡”,而被称为”链”。



上面描述的场景并不完善，因为客户端发来的报文访问的目标地址可能并不是本机，而是其他服务器，当本机的内核支持IP\_FORWARD时，我们可以将报文转发给其他服务器，所以，这个时候，我们就会提到iptables中的其他”关卡”，也就是其他”链”，他们就是“路由前PREROUTING”、”转发FORWARD”、”路由后POSTROUTING”

当我们启用了防火墙功能时，报文需要经过如下关卡，也就是说，根据实际情况的不同，报文经过”链”可能不同。如果报文需要转发，那么报文则不会经过input链发往用户空间，而是直接在内核空间中经过forward链和postrouting链转发出去的。

基本匹配条件：

“源**地址**Source **IP**，目标**地址** Destination **IP**”上述内容都可以作为基本匹配条件。

扩展匹配条件：

“源端口Source Port, 目标端口Destination Port”需要依赖对应的扩展模块。

处理动作：

处理动作在iptables中被称为target（这样说并不准确，我们暂且这样称呼），动作也可以分为基本动作和扩展动作。此处列出一些常用的动作，之后的文章会对它们进行详细的示例与总结：

ACCEPT：允许数据包通过。

DROP：直接丢弃数据包，不给任何回应信息，这时候客户端会感觉自己的请求泥牛入海了，过了超时时间才会有反应。

REJECT：拒绝数据包通过，必要时会给数据发送端一个响应的信息，客户端刚请求就会收到拒绝的信息。

SNAT：源地址转换，解决内网用户用同一个公网地址上网的问题。

MASQUERADE：是SNAT的一种特殊形式，适用于动态的、临时会变的ip上。

DNAT：目标地址转换。

REDIRECT：在本机做端口映射。

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，也就是说除了记录以外不对数据包做任何其他操作，仍然让下一条规则去匹配。

链的概念

iptables中为什么被称作”链”呢？防火墙的作用就在于对经过的报文匹配”规则”，然后执行对应的”动作”,所以，当报文经过这些关卡的时候，则必须匹配这个关卡上的规则每个经过这个”关卡”的报文，都要将这条”链”上的所有规则匹配一遍，如果有符合条件的规则，则执行规则对应的动作。

表的概念

我们对每个”链”上都放置了一串规则，但是这些规则有些很相似。我们把具有相同功能的规则的集合叫做”表”，所以说，不同功能的规则，我们可以放置在不同的表中进行管理，而iptables已经为我们定义了4种表，每种表对应了不同的功能，而我们定义的规则也都逃脱不了这4种功能的范围

①filter表：负责过滤功能，防火墙；内核模块：iptable\_filter

②nat表：network address translation，网络地址转换功能；内核模块：iptable\_nat

③mangle表：拆解报文，做出修改，并重新封装 的功能；iptable\_mangle

④raw表：关闭nat表上启用的连接追踪机制；iptable\_raw

也就是说，我们自定义的所有规则，都是这四种分类中的规则，或者说，所有规则都存在于这4张”表”中。

因为数据包经过一个”链”的时候，会将当前链的所有规则都匹配一遍，但是匹配时总归要有顺序，当他们处于同一条”链”时，执行的优先级如下。优先级次序（由高而低）：

raw –> mangle –> nat –> filter

表链关系

让我们来看看，每个”关卡”都有哪些能力，或者说，让我们看看每个”链”上的规则都存在于哪些”表”中。（没有先后顺序）

PREROUTING 的规则可以存在于：raw表，mangle表，nat表。

INPUT 的规则可以存在于：mangle表，filter表，（centos7中还有nat表，centos6中没有）。

FORWARD 的规则可以存在于：mangle表，filter表。

OUTPUT 的规则可以存在于：raw表mangle表，nat表，filter表。

POSTROUTING 的规则可以存在于：mangle表，nat表。

我们在实际的使用过程中，往往是通过”表”作为操作入口，对规则进行定义的，所以下面将各”表”与”链”的关系罗列出来。

表（功能）<–> 链（钩子）：

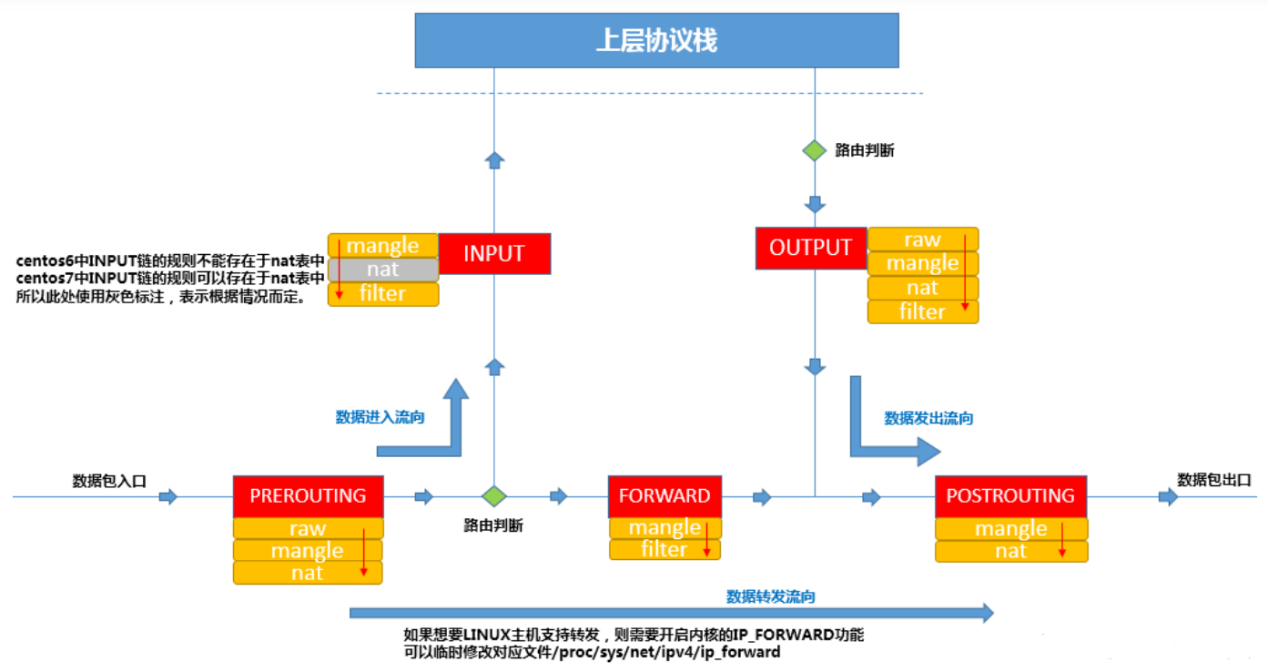
raw 表中的规则可以被哪些链使用：PREROUTING，OUTPUT

mangle 表中的规则可以被哪些链使用：PREROUTING，INPUT，FORWARD，OUTPUT，POSTROUTING

nat 表中的规则可以被哪些链使用：PREROUTING，OUTPUT，POSTROUTING（centos7中还有INPUT，centos6中没有）

filter 表中的规则可以被哪些链使用：INPUT，FORWARD，OUTPUT

数据经过防火墙的流程



# **[iptables操作：规则查询](https://www.zsythink.net/archives/1493)**

实际操作iptables的过程中，是以”表”作为操作入口的。当我们定义iptables规则时，所做的操作其实类似于”增删改查”

之前的文章中，我们已经总结过，iptables为我们预定义了4张表，它们分别是raw表、mangle表、nat表、filter表，不同的表拥有不同的功能。filter负责过滤功能，比如允许哪些IP地址访问，拒绝哪些IP地址访问，允许访问哪些端口，禁止访问哪些端口，filter表会根据我们定义的规则进行过滤，filter表应该是我们最常用到的表了，所以此处，我们以filter表为例，开始学习怎样实际操作iptables。

查看filter中的规则：iptables -t filter -L

（使用-t选项，指定要操作的表； -L选项的意思是，列出规则）



查看其它表中的规则，示例如下。

iptables -t raw -L iptables -t mangle -L iptables -t nat -L

当没有使用-t选项指定表时，默认为操作filter表，即iptables -L表示列出filter表中的所有规则

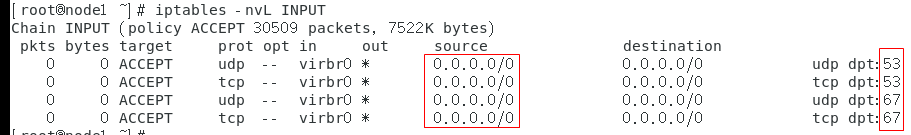
只查看filter表中INPUT链的规则：iptables -L INPUT

可以使用-v选项，查看出更多的、更详细的信息: iptables -vL INPUT



表中的每个链的后面都有一个括号，括号里面有一些信息。INPUT链后面的括号中包含policy ACCEPT ，0 packets，0bytes 三部分。policy表示当前链的默认策略，默认ACCEPT说明大家都能通过这个关卡，在配置INPUT链的具体规则时只要将需要拒绝的请求配置到规则中（黑名单）。packets表示当前链（上例为INPUT链）默认策略匹配到的包的数量。bytes表示当前链默认策略匹配到的所有包的大小总和。

可以使用-n选项，表示不对IP地址进行名称反解，直接显示IP地址。上图中的源地址与目标地址都为anywhere，是因为iptables默认为我们进行了名称解析，但是在规则非常多的情况下如果进行名称解析，效率会比较低。 iptables -nvL INPUT



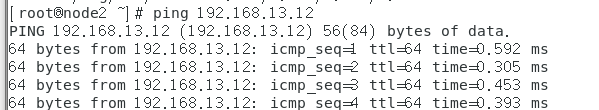
使用--line-numbers即可显示规则的编号：iptables --line-numbers -nvL INPUT

当被匹配到的包达到一定数量时，计数器会自动将匹配到的包的大小转换为可读性较高的单位

如果你想要查看精确的计数值，而不是经过可读性优化过的计数值，那么你可以使用-x选项，表示显示精确的计数值

# **[iptables操作：规则管理](https://www.zsythink.net/archives/1517)**

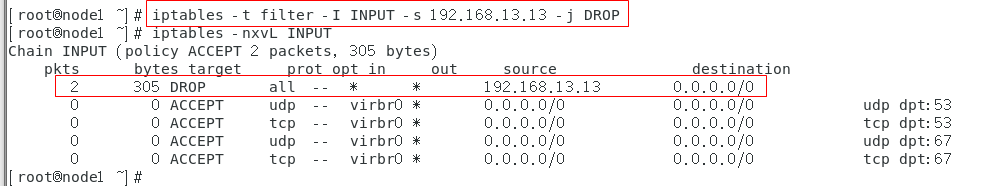
先用node2(192.168.13.13) ping一下node1(192.168.13.12)



## **（一）增加规则**

那么此处，我们就在node1(192.168.13.12)上配置一条规则，拒绝node2(192.168.13.13) 上的所有报文访问当前机器。这条规则中，报文的”源地址为(192.168.13.13) ″则属于匹配条件，如果报文来自”192.168.13.13″，则表示满足匹配条件，而”拒绝”这个报文，就属于对应的动作。

iptables -t filter -I INPUT -s 192.168.13.13 -j DROP



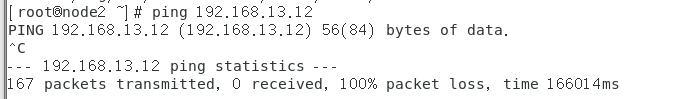
-t选项指定了要操作的表，此处指定操作filter表，不使用-t指定时，默认filter表。

-I选项，指明将”规则”插入至哪个链中

-s选项，指明”匹配条件”中的”源地址”

-j选项，指明当”匹配条件”被满足时，所对应的动作

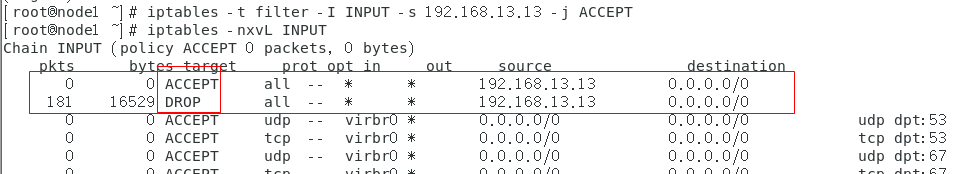
node1加完规则之后，再用node2尝试ping 192.168.13.12， 无法连接

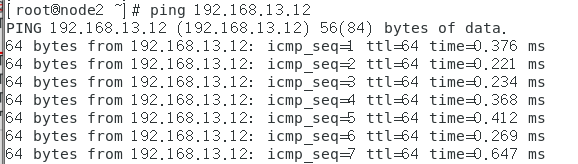


如果此时，我们在这条规则之后再配置一条规则接受node2的请求：

iptables -t filter -I INPUT -s 192.168.13.13 -j ACCEPT

注-I INPUT表示在链的头部添加， -A INPUT表示在链的尾部添加





先ACCEPT再DROP 能ping通 ； 如果先DROP再ACCEPT的话 不能ping通

结论：如果报文被多个规则匹配，只有第一条生效，如果报文已经被前面的规则匹配到，iptables则会对报文执行对应的动作，报文就已经被放行了，即使后面的规则也能匹配到当前报文，很有可能也没有机会再对报文执行相应的动作了。

之前讲过，使用–line-number选项可以列出规则的序号；在添加规则时，指定新增规则的编号，这样我们就能在任意位置插入规则了,新增的规则的编号为2

iptables -t filter -I INPUT 2 -s 192.168.13.13 -j DROP

## **（二）删除规则**

1、按照规则序号删除规则：

命令语法：iptables -t 表名 -D 链名 规则序号

（使用--line-numbers可显示规则的编号）

示例：iptables -t filter -D INPUT 3

2、按照具体的匹配条件与动作删除规则：

命令语法：iptables -t 表名 -D 链名 匹配条件 -j 动作

示例：iptables -t filter -D INPUT -s 192.168.13.13 -j DROP

表示删除filter表中INPUT链中源地址为192.168.13.13并且动作为DROP的规则。

1. 删除指定表中某条链中的所有规则的命令：

命令语法：iptables -t 表名 -F 链名

示例：iptables -t filter -F INPUT

-F选项为flush之意，即冲刷指定的链，即删除指定链中的所有规则

不指定链名，只指定表名即可删除表中的所有规则：iptables -t 表名 -F