

# GNU/Linux iptables



# iptables

防火墙 (Firewall):

属于建筑术语，在建筑房屋的时候有些墙面被设计成具有一定的防火能力，这样在发生火灾的时候，可以对火势的蔓延起到一定的阻隔作用



# iptables

防火墙 (Firewall):

计算机网络中的防火墙通常连接两个网络，是外部 Internet 和内部网络之间的交汇点，同时也是一道屏障

主要功能：实施安全策略、过滤传输数据、记录 Internet 活动、IP 地址转换、保护内部网络信息

# iptables

防火墙类型：

1. 包过滤防火墙
2. 代理防火墙
3. 状态检测防火墙



# iptables

## 包过滤防火墙：

包过滤防火墙对通过它的每一个数据包，根据事先制订好的规则，对它的源地址、目的地址以及相应的端口进行判断，把不合规则的数据包都过滤掉。

。



# iptables

包过滤防火墙：

优点：

处理来速度快，对用户透明

缺点：

不能实现用户级认证

日志记录不完善

无法防御 IP 欺骗

对特殊协议不适用



# iptables

代理 (Proxy) 防火墙：

代理也称为应用网关防火墙，核心是代理技术，即代替客户处理对服务器连接请求。

优点：

安全性高

过滤规则灵活

详细的日志记录



# iptables

代理 (Proxy) 防火墙：

缺点：

- 处理速度慢

- 对用户不透明

- 代理服务类型有限制





# iptables

状态检测防火墙：

又称为动态包过滤防火墙。其主要特点在于

1) 在防火墙的核心部分建立状态连接表，并将进出网络的数据当成一个个的会话，利用状态表跟踪每个会话状态。

2) 防火墙首先根据规则表来判断是否允许这个数据包通过，如果不符合规则就立即丢弃；如果符合规则，再将当前的数据包和状态信息，与前一时刻的数据包和状态信息进行比较，然后根据比较结果决定数据包是否能够通过防火墙。

# iptables

防火墙结构：

屏蔽路由器

双穴主机

屏蔽主机防火墙

屏蔽子网防火墙

其它结构



# iptables

屏蔽路由器（包过滤）

优点：

处理速度快

费用低

对用户透明



# iptables

屏蔽路由器（包过滤）

缺点：

维护困难

只能阻止较少的 IP 欺骗

无用户认证

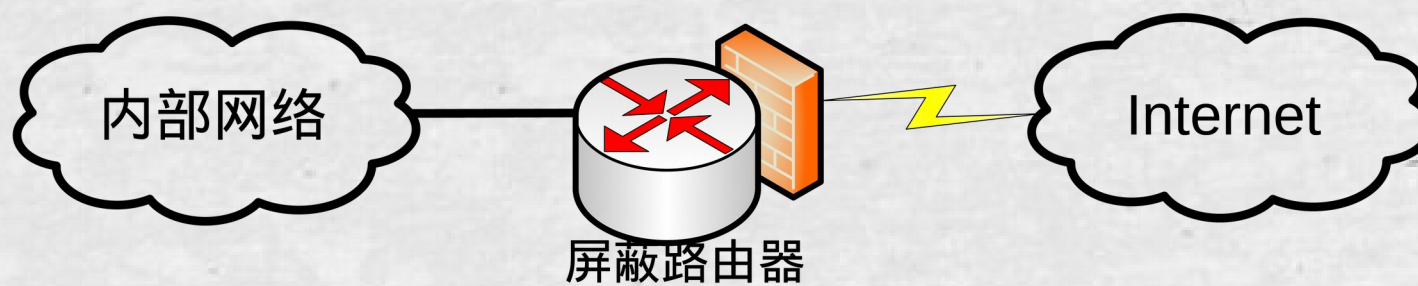
日志功能有限，

当规则过多时，处理速度及吞吐速度急速下降，  
无法对信息全面控制



# iptables

屏蔽路由器（包过滤）

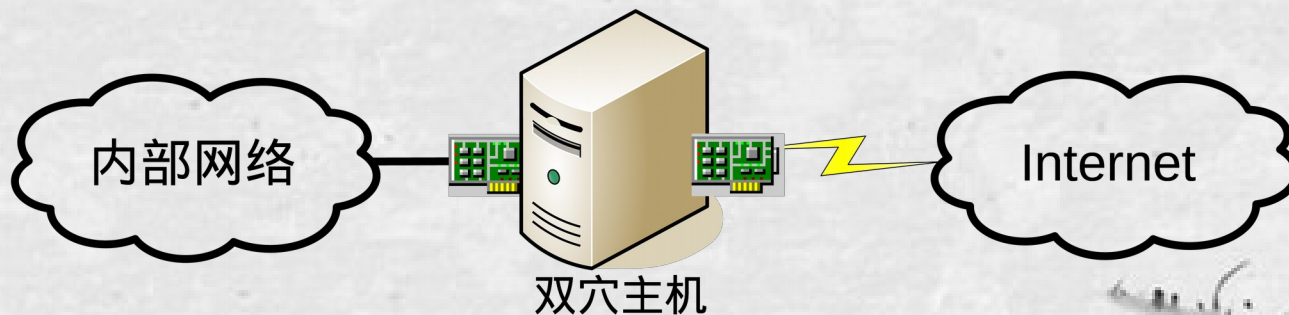


# iptables

双穴主机（双宿网关）

优点：安全性比屏蔽路由器高

缺点：入侵者一旦得到双穴主机的访问权，内部网络就会被入侵，因此需具有强大的身份认证系统，才可以阻挡来自外部的不可信网络的非法入侵。

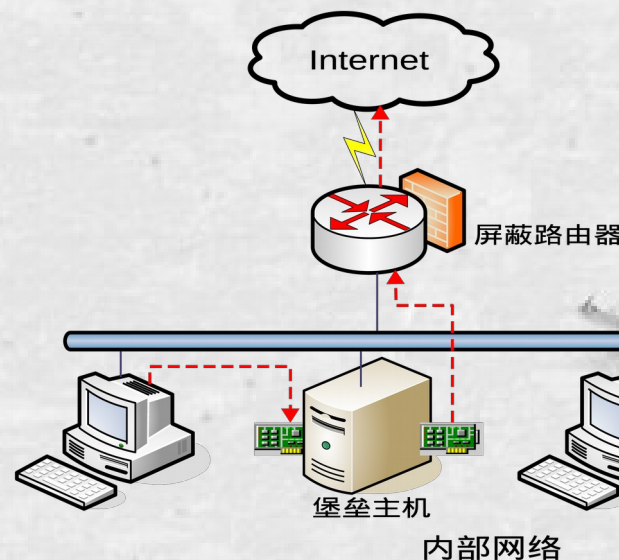


# iptables

## 屏蔽主机防火墙

优点：实现了网络层安全（包过滤）和应用层安全（代理），因此安全等级比屏蔽路由器高

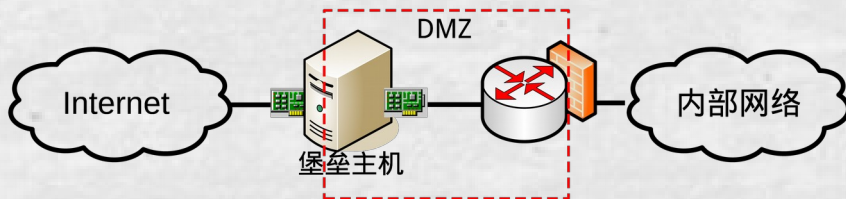
缺点：堡垒主机可能被绕过，堡垒主机与其它内部主机间没有任何保护网络安全的东西存在，一旦被攻破，内网就将暴露



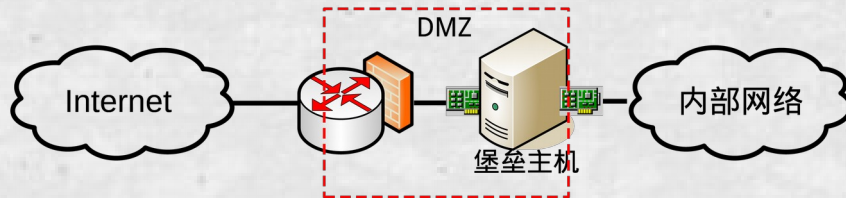


# iptables

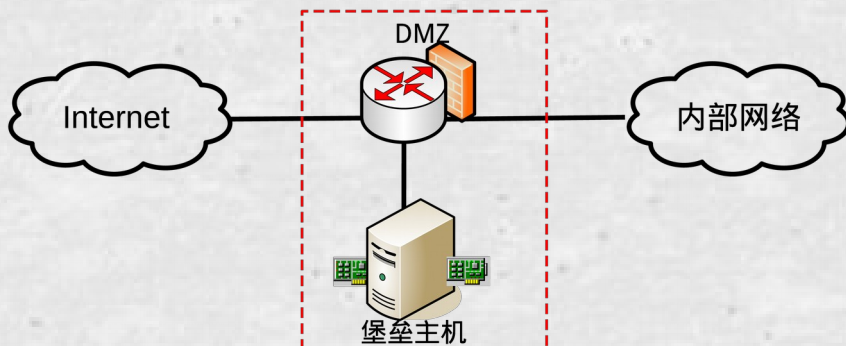
## 其他结构



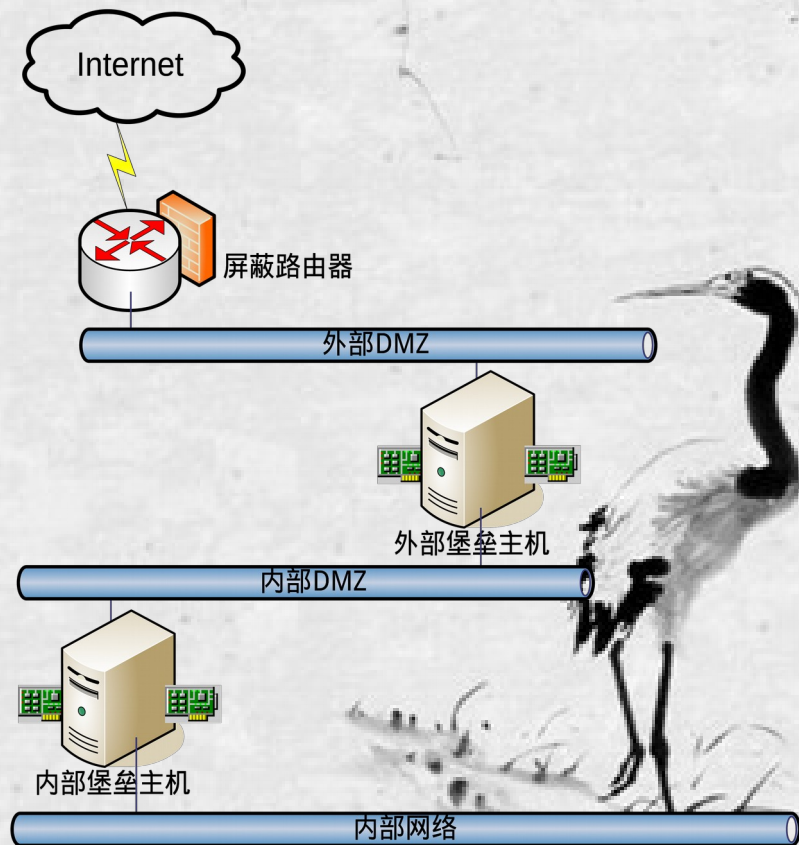
(a) 合并外部路由器与堡垒主机



(b) 一个堡垒主机和一个DMZ结构



(c) 合并内外路由器





# iptables

## Linux 防火墙

Linux 防火墙采用包过滤技术，总体看防火墙发展经历了四个发展阶段

静态防火墙：

ipfwadm(kernel 2.0)

ipchains(kernel 2.2)

iptables(kernel 2.4 later)

动态防火墙

firewalld(RHEL7 新纳入的防火墙)



# iptables

## Linux 防火墙

### 静态防火墙：

需要自行书写规则及相关信息，每次改变后都需要重新启动防火墙

### 动态防火墙

在更改防火墙规则后，不需要重新加载相关服务和模块，达到改变及应用的效果



# iptables

## Linux 防火墙

firewalld 看起来是一个非常新的防火墙，实际上仍然采用了 netfilter/iptables 的底层架构。虽然在应用上变化非常大，但底层实质意义上没有太多的改变。



# iptables

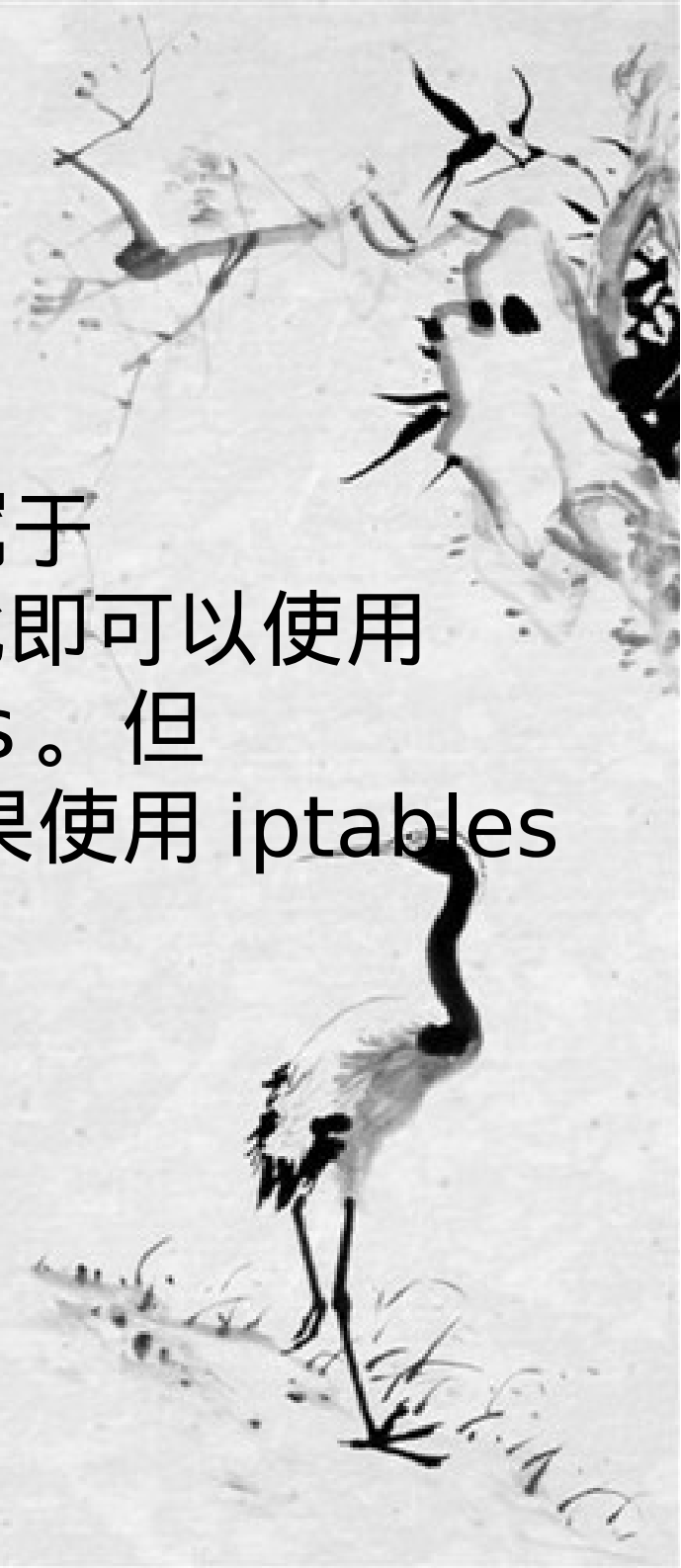
## iptables

虽然 firewalld 推荐使用，但仍属于 inetfilter/iptables 作为底层。因此即可以使用 firewalld 也可以继续使用 iptables。但 firewalld 与 iptables 有冲突。如果使用 iptables 就必须关闭 firewalld。

```
#systemctl unmask iptables
```

```
#systemctl mask firewalld
```

```
#systemctl start iptables
```



# iptables

## iptables

Netfilter 指定 IPTABLES 来编写防火墙其主要功能有：

1. 状态包过滤（ 连接跟踪 ）
2. 各种网络地址翻译
3. 灵活，非常容易扩展的应急机制
4. 大量的增强型补丁

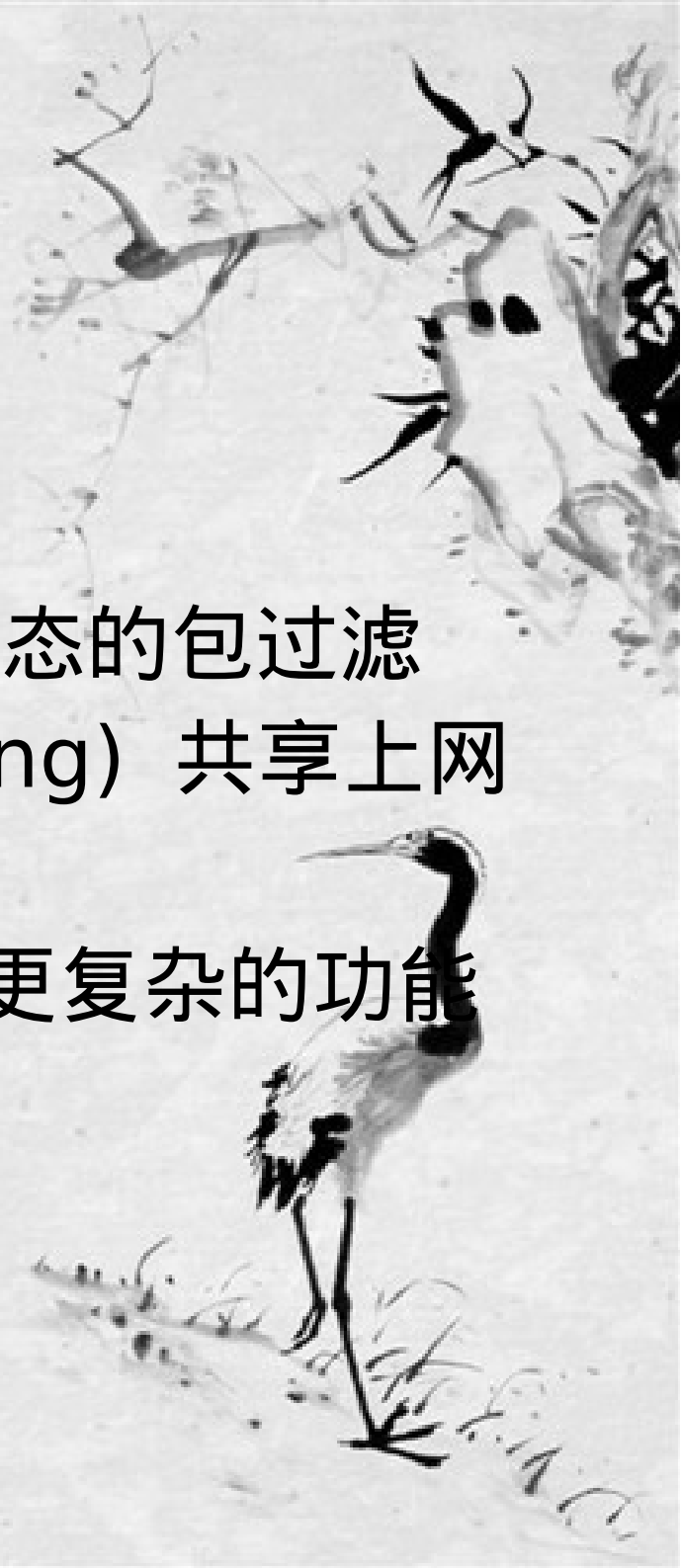


# iptables

## iptables

主要用途有：

1. 建立 internet 防火墙和基于状态的包过滤
2. 用 NAT 和伪装 (masquerading) 共享上网
3. 用 NAT 实现透明代理
4. 用修改 IP 包头的 ToS 来实现更复杂的功能  
等等 .....



# iptables

iptables

netfilter/iptables 可以配置有状态的防火墙

有状态防火墙主要是能够指定记住为发送或者接收信息包所建立的连接的状态。防火墙可以从信息包的连接跟踪状态获得该信息。在决定性的信息包的过滤的时候，防火墙所使用的这些状态信息可以增加它的效率和速度。状态有 4 种：


# iptables



## iptables

1. ESTABLISHED: 这个状态是指出这个信息包属于已经建立的连接。这个连接一直用于发送很接收信息包并且完全有效

2. INVALID: 这个状态指出该信息包与任何已经知道的流或者连接都不相关联。它可能饱含错误的数据或者头





# iptables

## iptables

3.NEW: 这个状态指出该信息包已经或者准备启动新的连接，或者它与尚未用于发送和接受的信息包的连接相关联

4.RELATED: 这个状态指出该信息包正在启动新的连接，以及它与已经建立的连接相关联



# iptables

## iptables

Netfilter/iptables 是一个套件，由两个部分组成：

1.Netfilter---- 内核空间 :netfilter 组件也称为内核空间，用来存储策略及过滤

2.iptables---- 用户空间：

iptables 组件是一种工具 用来编写策略。



# iptables

## iptables 表与链

1)filter( 默认 )

filter 这个表主要有三种内置链组成：

INPUT 链处理送入本机的流入报文

FORWARD 链处理经过本系统的报文

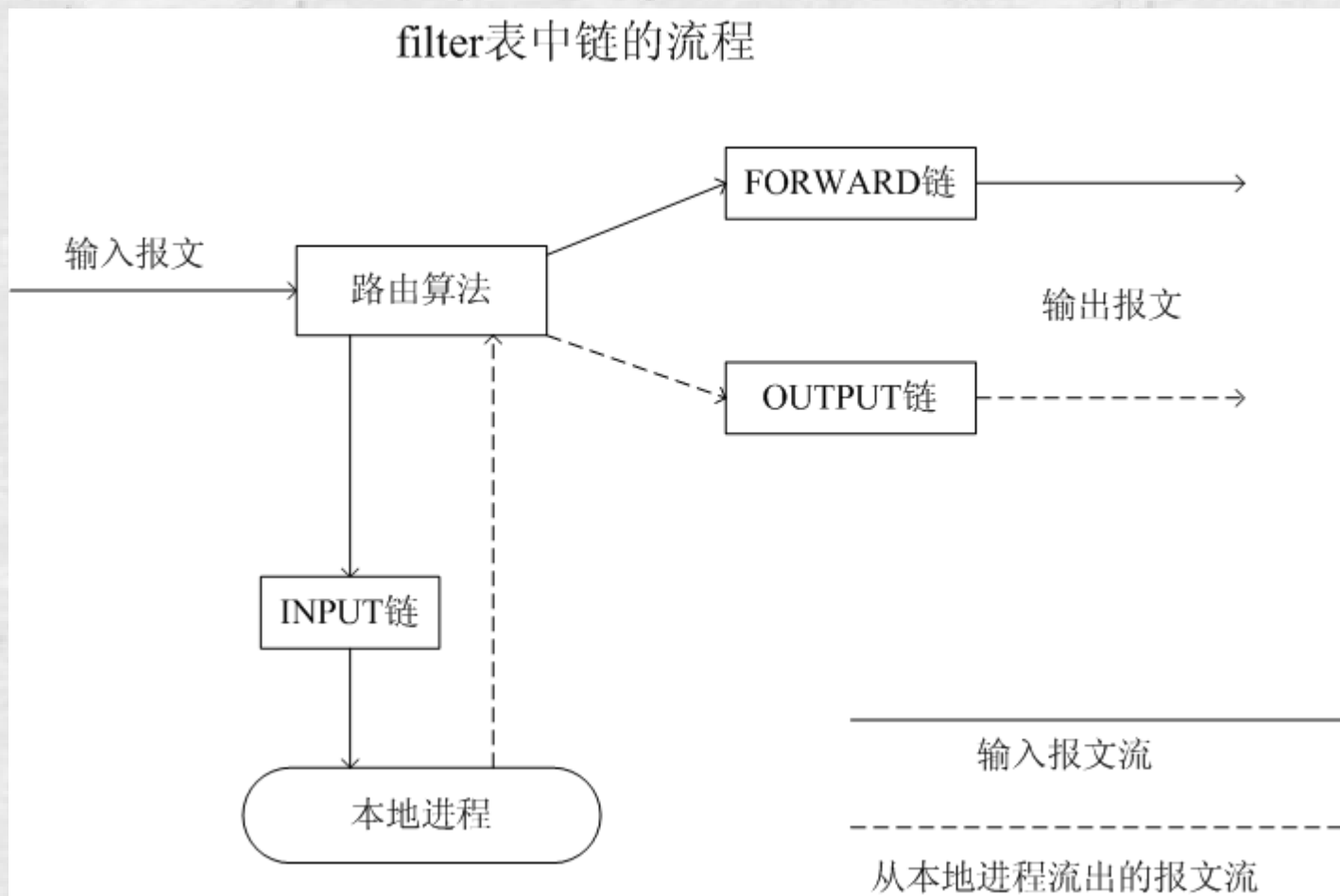
OUTPUT 链处理从本地发出的报文



# iptables

## iptables 表与链

### 1)filter 中链工作流程



# iptables

iptables 表与链

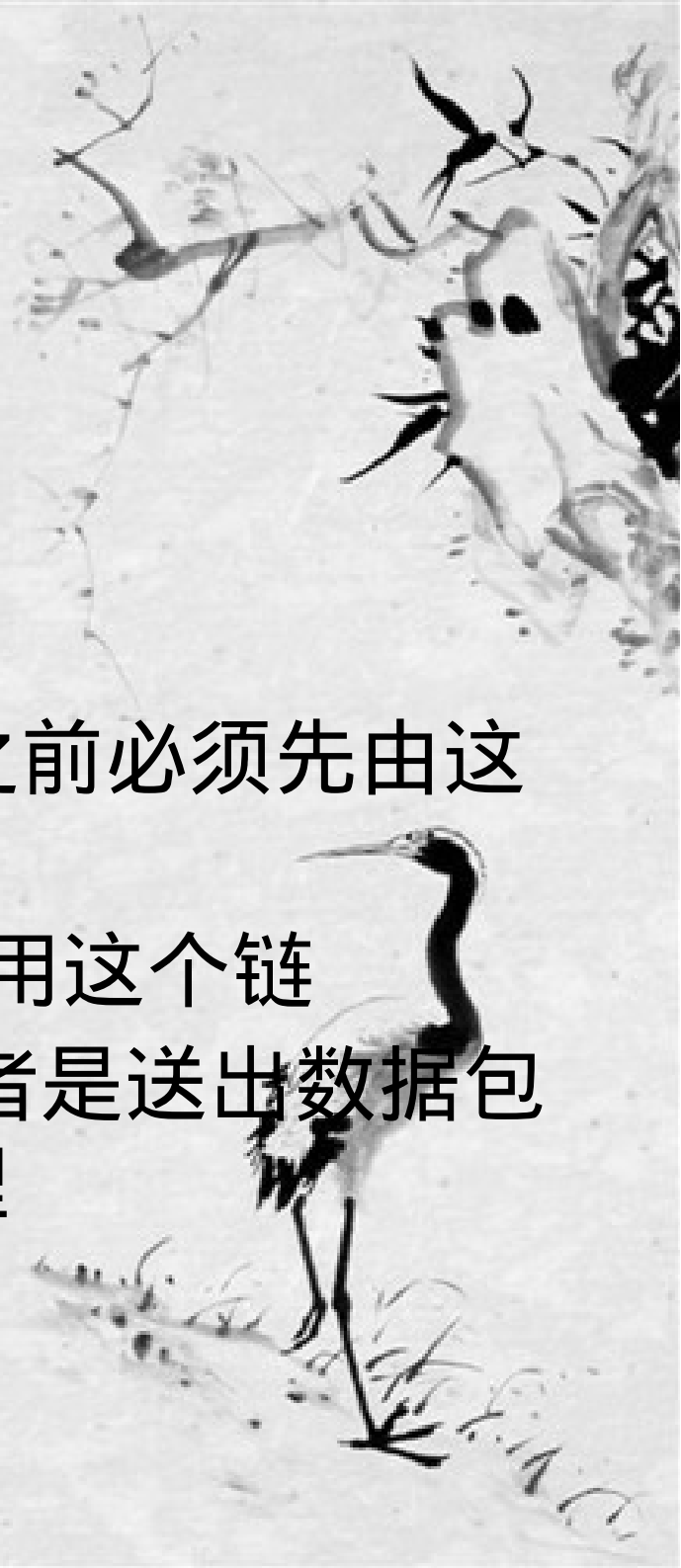
2) nat

nat 表也有三种内置链：

PREROUTING 链：在路由决策之前必须先由这个链来处理，以完成地址转换

OUTPUT 链：从本地发送报文使用这个链

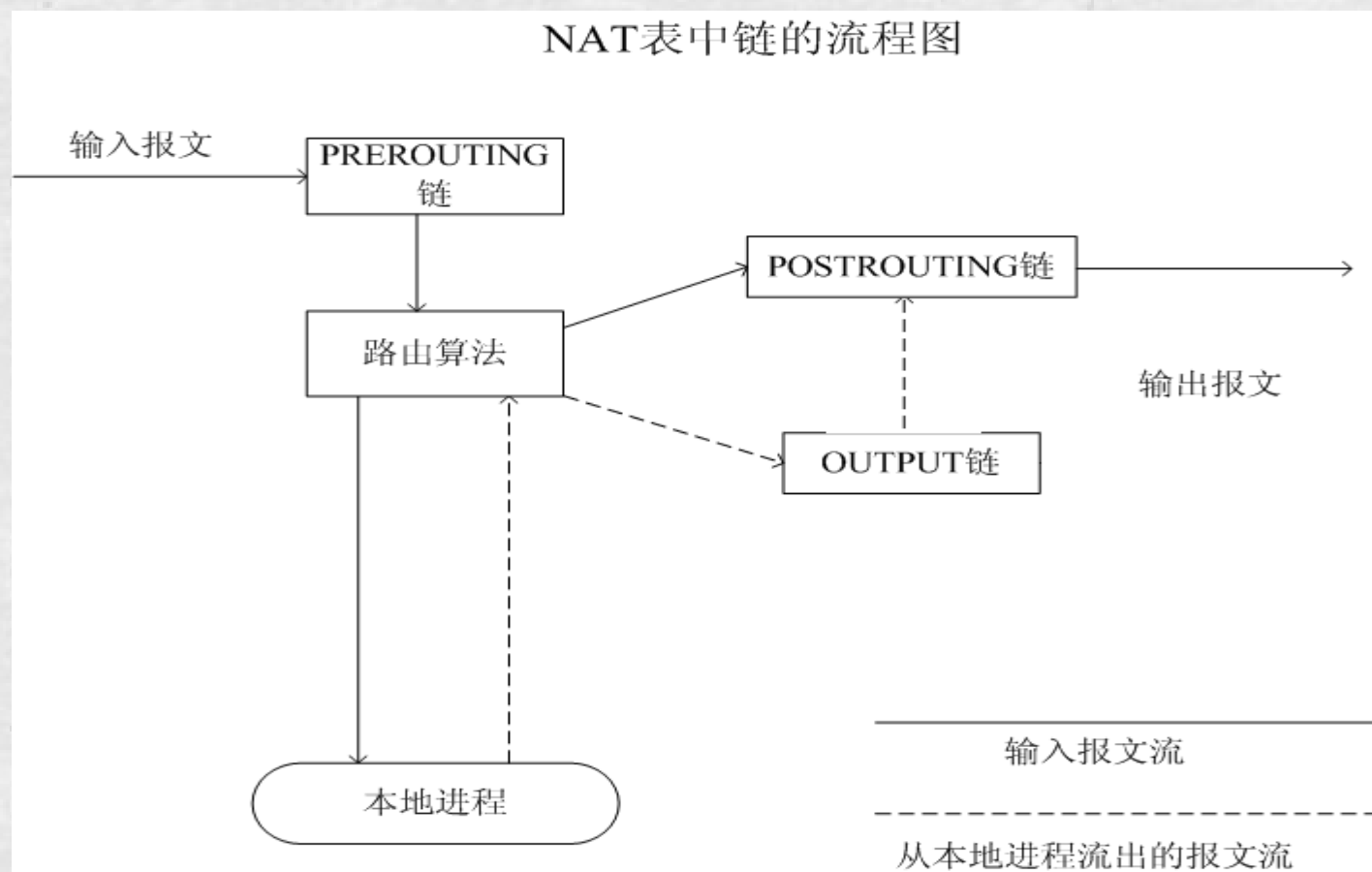
POSTROUTING：如果是转发或者是送出数据包，则由 POSTROUTING 来进行处理



# iptables

## iptables 表与链

### 2)nat 表中链的工作流程



# iptables

iptables 表与链

3)mangle 表

由两种链组成：

PREROUTING 链：主要用于在路由决策之前对流入报文的过滤

OUTPUT 链：主要用于在路由决策之前对从本地送出的报文过滤

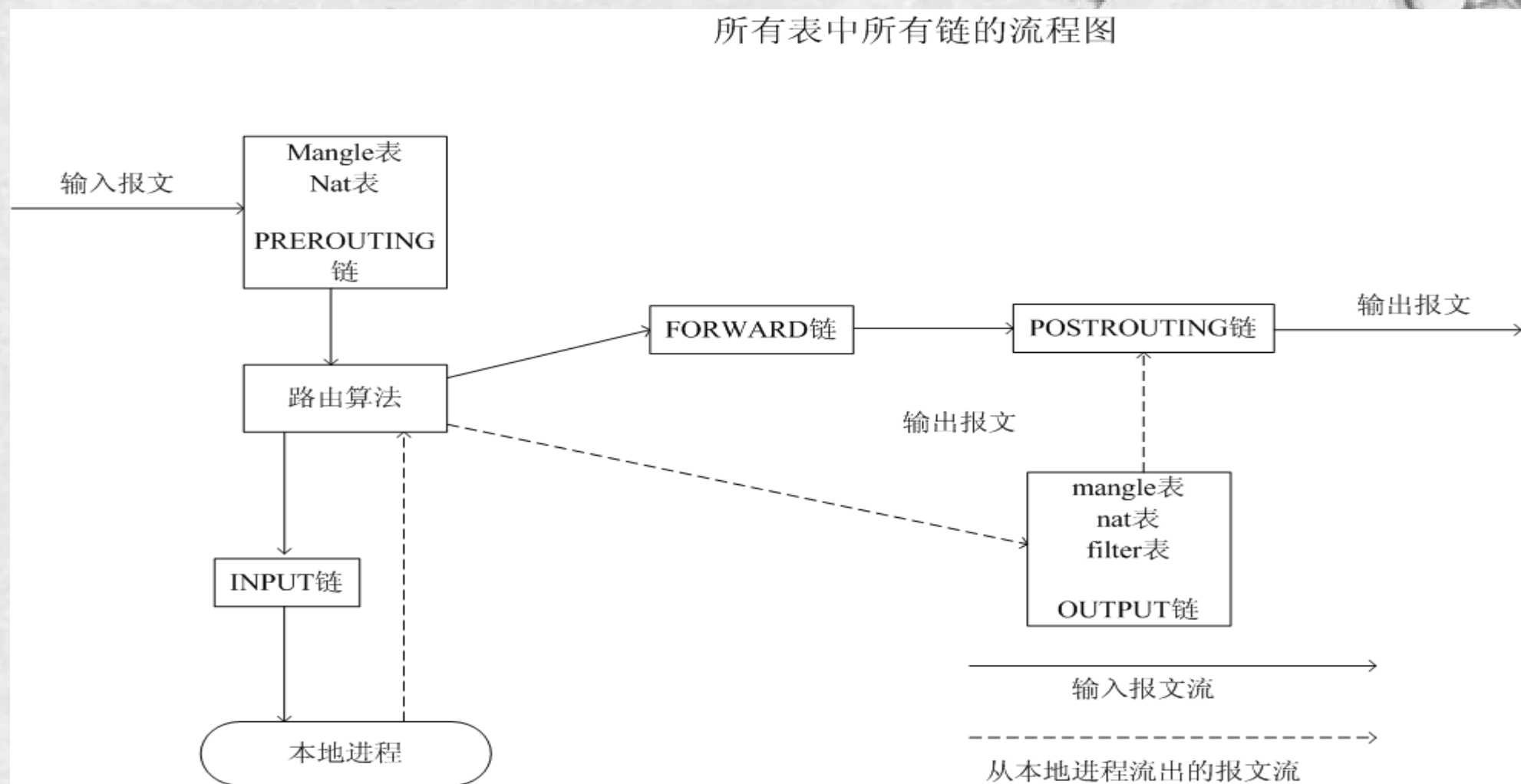
本课程对 mangle 表不作探讨



# iptables

## iptables 表与链的工作流程

所有表中所有链的流程图





# iptables

iptables 语法格式

#iptables [ 表 ] [ 命令 ] [ 链名 ] [ 规则 ] [ 动作 ]

表：

filter/nat/mangle

链：

INPUT/OUTPUT/FORWARD/POSTROUTING/PR  
EROUTING



# iptables

iptables 语法格式

动作：

ACCEPT: 允许数据包通过

DROP: 拒绝数据包通过，不会告诉发送者数据包被废弃



# iptables

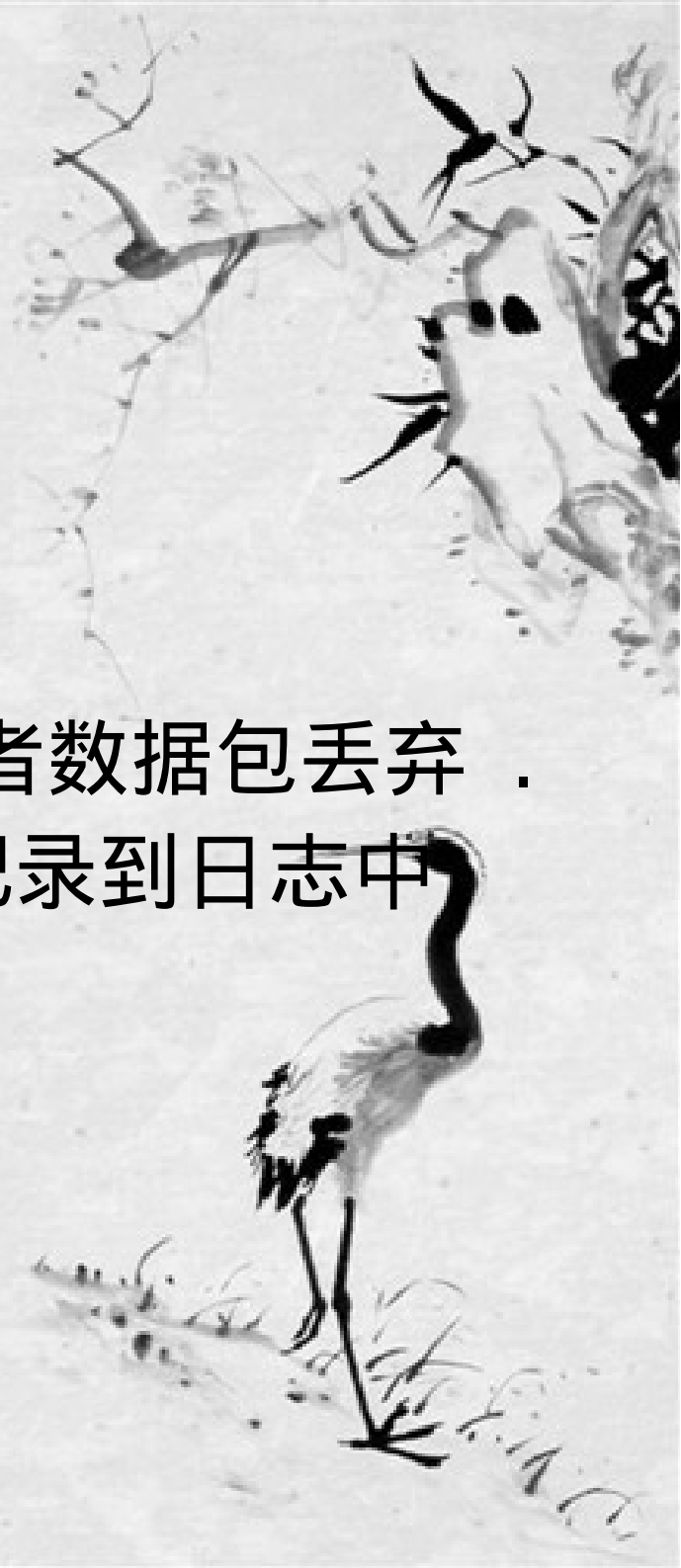
iptables 语法格式

IPTABLES 扩展动作

REJECT: 丢弃数据包并且告诉发送者数据包丢弃 .

LOG: 表示与数据包有关的信息被记录到日志中

ToS: 改写数据包的 Tos



# iptables

iptables 语法格式

命令：

-t: 指定表名

-A: 在指定链最后加入一条新规则

-D: 删除链中指定规则

-R: 替换链中一条规则



# iptables

iptables 语法格式

命令：

-I: 在指定规则前插入一条规则

-L: 列出指定链的规则

-F: 清除指定链 / 表中的规则

-N: 自定义链



# iptables

iptables 语法格式

命令：

-X: 删除自定义链

-P: 设定默认策略

-C: 检查给定的数据包与指定链接规则是否相匹配

-Z: 清空指定链中的所有计数



# iptables

iptables 语法格式

命令：

-h:iptables 帮助



# iptables

iptables 语法格式  
规则匹配

-p [!] protocol: 指定匹配的协议  
(tcp/udp/icmp..)

-s [!] address[/mask]: 指定匹配的源地址

--sport [!] port[:port]: 指定源端口或端口范围

-d [!] address[/mask]: 指定匹配的目的地地址





# iptables

iptables 语法格式  
规则匹配

--dport [!] port[:port]: 指定目的端口或端口范围

--icmp-type [!] typename: 指定 icmp 的类型，  
可使用 iptables -p icmp -h 来查看 icmp 类型

-i [!] interface\_name[+]: 指定入流的网卡接口  
或类型，如 eth+ (仅限  
INPUT/FORWARD/PREROUTING)

# iptables

iptables 语法格式  
扩展规则匹配

-m mac --mac-source [!] addr: 匹配源 MAC 地址

-m limit: 限定包数量

-m mac --mimit rate: 匹配包速率

-m mac -limit-burst: 显示宽限数



# iptables

iptables 语法格式

扩展规则匹配

-m multiport --source-port

port1,port2[,port3—15]: 指定 15 个以内的源端

□

-m multiport --destination-port

port1[,port2...15]: 指定 15 个以内的目的端口



# iptables

iptables 语法格式

动作：

ACCEPT: 允许通过

DROP: 拒绝，并不告知对方

REJECT: 拒绝，告知对反

MASQUERAD [ --to-port port [ -port]]: 伪装



# iptables

iptables 语法格式

动作：

LOG

[--log-level level]: 将匹配的信息记录到日志

[ --log-prefix string]: 将 string 作为前缀记录到日志中，以便于过滤及查询

--log-tcp-sequence: 记录 tcp 序列号



# iptables

iptables 语法格式

动作：

--log-ip-options: 记录有关协议的选项信息

DNAT --to-destination ipaddress[:port-port]:NAT 转换，将源地址转换为目的地址（用于 PREROUTING）

SNAT --to-source ipaddress[:port-port] 将目的地址转换为源地址（用于 POSTROUTING）

# iptables

## iptables 使用

### 1. 链的基本使用

1) 清除预设表 (filter) 中的所有规则链中的规则

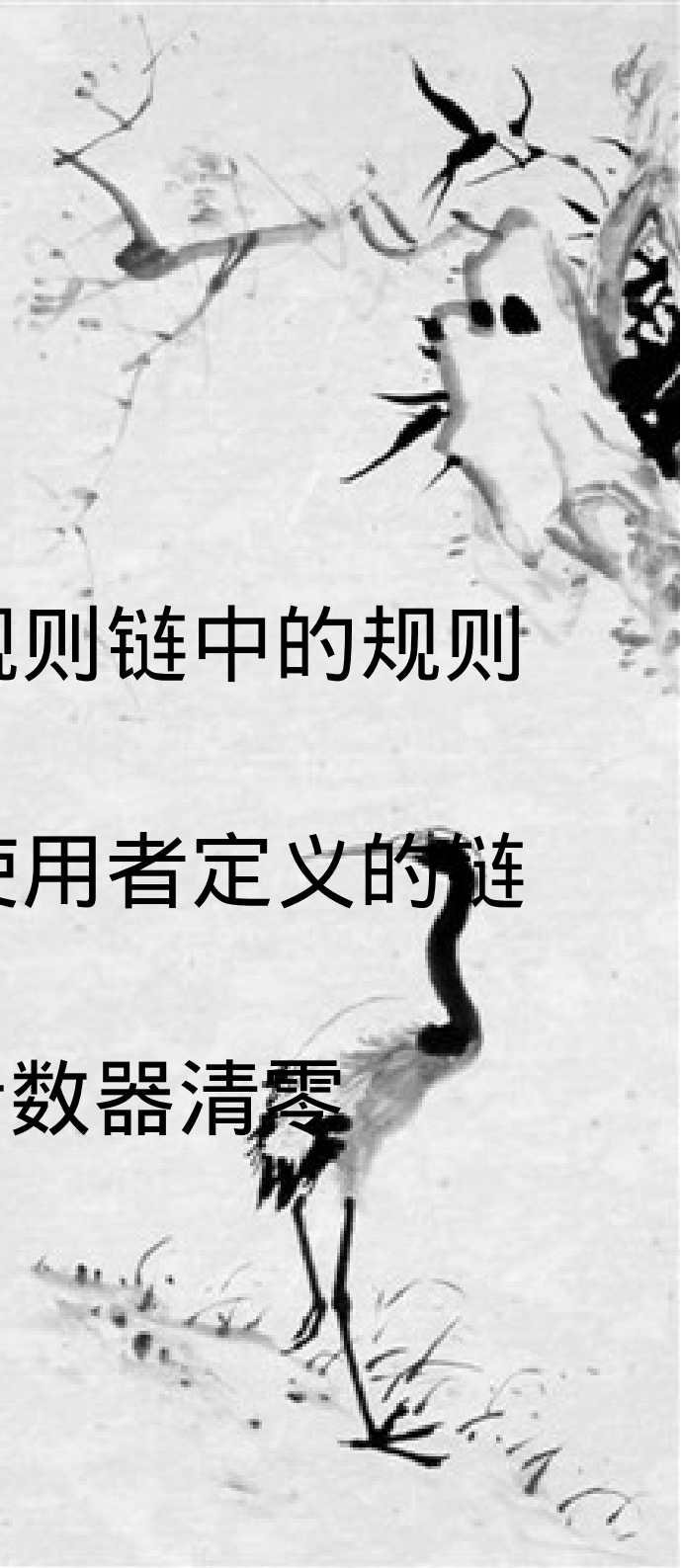
#iptables -F

2) 清除预设表 (filter) 中的所有使用者定义的链

#iptables -X

3) 将指定链中所有规则的包字节计数器清零

#iptables -Z



# iptables

iptables 使用

2. 设置链的默认策略 .

```
#iptables -P INPUT DROP
```

```
#iptables -P OUTPUT DROP
```

```
#iptables -P FORWARD DROP
```





# iptables

iptables 使用

3. 列出表 / 链的所有规则 .

1) 列出 filter 表中的规则

#iptables -L

2) 列出 nat 表中的规则

#iptables -t nat -L

3) 列出 mangle 表中的规则

#iptables -t mangle -L



# iptables

## iptables 使用

4. 向链中添加规则 . 以下的是说明开放哪些端口

1) 在 INPUT 链尾处添加一条允许数据包进入 lo 设备

```
#iptables -A INPUT -i lo -j ACCEPT
```

2) 在 OUTPUT 链尾处添加一条允许通过 lo 设备传出数据包

```
#iptables -A OUTPUT -o lo -j ACCEPT
```

# iptables

## iptables 使用

4. 向链中添加规则 . 以下的是说明开放哪些端口

3) 在 INPUT 链尾处添加一条允许数据包进入 eth0 设备

```
#iptables -A INPUT -i eth0 -j ACCEPT
```

4) 在 OUTPUT 链尾处添加一条允许通过 eth0 设备传出数据包

```
#iptables -A OUTPUT -o eth0 -j ACCEPT
```

# iptables

## iptables 使用

4. 向链中添加规则 . 以下的是说明开放哪些端口

5) 在 INPUT 链尾处添加一条允许数据包通过 eth1 设备进行转发

```
#iptables -A INPUT -i eth1 -j ACCEPT
```

6) 在 OUTPUT 链尾处添加一条允许通过 eth1 设备转发数据包

```
#iptables -A OUTPUT -o eth1 -j ACCEPT
```

# iptables

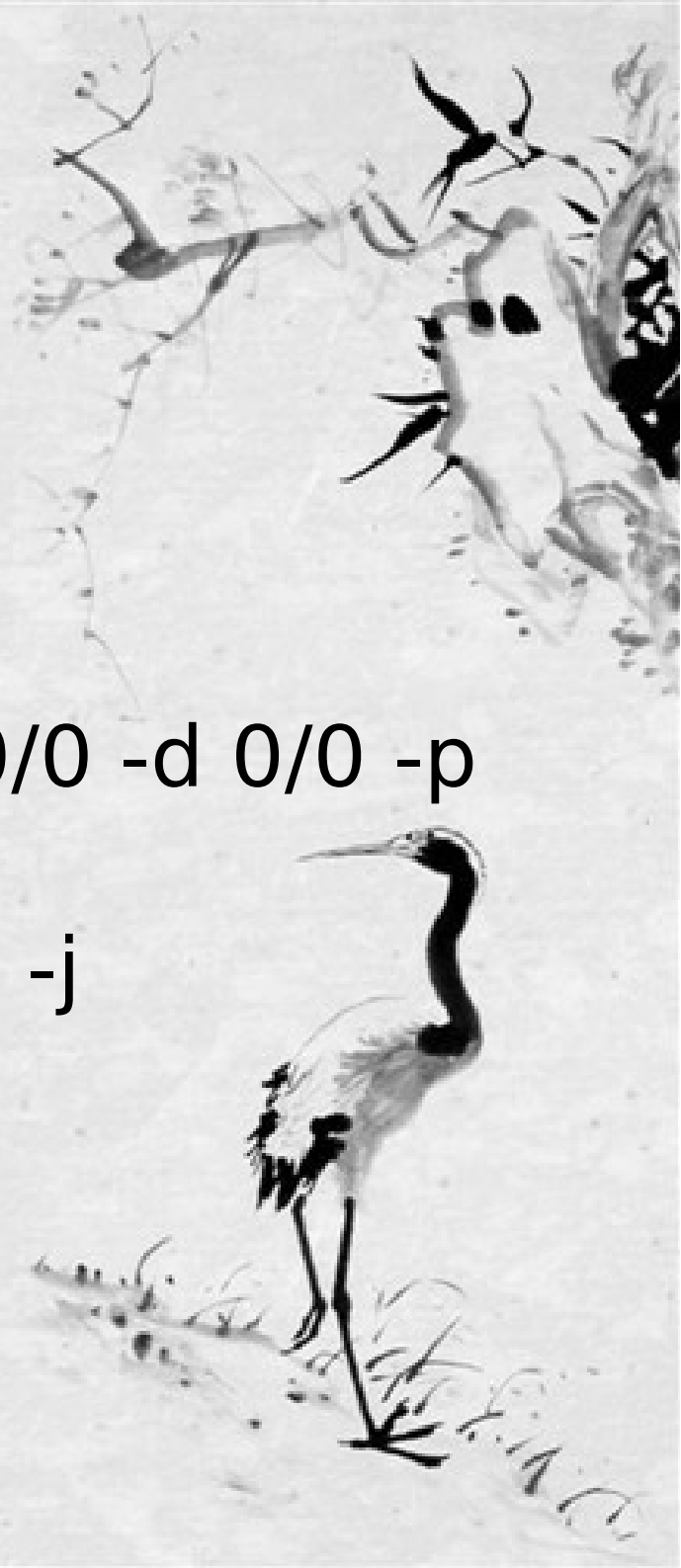
iptables 使用

5. 使用用户自定义链

```
#iptables -N custom_name
```

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

```
#iptables -A INPUT -s 0/0 -d 0/0 -j  
custom_name
```



# iptables

iptables 使用

6. 指定基本规则匹配

1) 指定匹配协议

```
#iptables -A INPUT -p tcp
```

2) 指定除了此协议之外的匹配

```
#iptables -A INPUT -p ! tcp
```

3) 指定匹配的主机

```
#iptables -A INPUT -s 192.168.1.1
```



# iptables

iptables 使用

6. 指定基本规则匹配

4) 指定匹配的网络段

```
#iptables -A INPUT -s 192.168.1.0/24
```

5) 指定除了某个主机以外的匹配

```
#iptables -A INPUT -s ! 192.168.1.1
```

6) 指定除了某个网络段的匹配

```
#iptables -A INPUT -s ! 192.168.1.0/24
```



# iptables

7) 指定某个匹配的接口

```
# iptables -A INPUT -i eth0
```

8) 指定某个匹配的接口类型

```
#iptables -A INPUT -i ppp+
```

9) 指定某个匹配的端口

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

```
#iptables -A INPUT -p tcp --sport 80
```

10) 匹配除了某个端口之外的端口

```
#iptables -A INPUT -p tcp --sport !22
```





# iptables

11) 匹配某个端口范围

```
#iptables -A INPUT -p tcp --sport 20:80
```

12) 匹配 ICMP 端口和 ICMP 类型

```
#iptables -A INPUT -p icmp --icmp-type 8
```



# iptables

## 示例

### 1. 指定 IP 碎片

```
#iptables -A INPUT -p tcp -f -s  
192.168.1.0/24 -d 0/0 --dport www -j  
ACCEPT
```

### 2. 防止 DoS 攻击

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



# iptables

## 示例

### 1. 指定 IP 碎片

```
#iptables -A INPUT -p tcp -f -s  
192.168.1.0/24 -d 0/0 -dport www -j ACCEPT
```

### 2. 防止 DoS 攻击，每分钟 3 个 icmp 包，最大不 烧过 6 个

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

# iptables

7. 设置扩展的规则匹配 . 主要参数 :-m

1) 匹配多个源端口

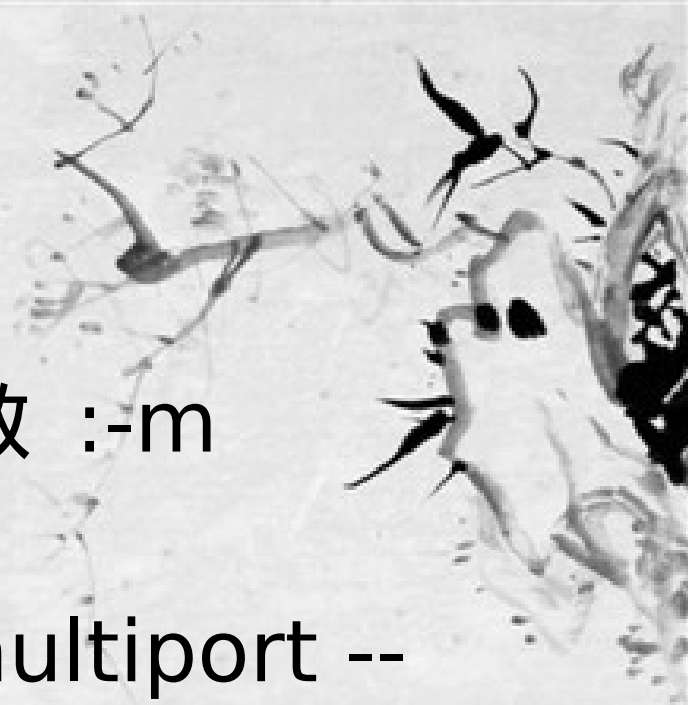
```
#iptables -A INPUT -p tcp -m multiport --  
source-port 20,21,25,53,80,110
```

2) 匹配多个目的端口

```
#iptables -A INPUT -p tcp -m multiport --  
destination-port 20,21,25,53,80,110
```

3) 同时匹配源和目的端口

```
#iptables -A INPUT -p tcp -m multiport  
--port 20,21,25,53,80,110
```



# iptables

## 8. TCP 标志

SYN: 同步标记

ACK: 应答标记

RST: 复位标记

URG: 紧急标记

FIN: 结束标记

PSH: 强制推送标记

.....



# iptables

## 8. TCP 标志

TCP 的非法标记：

非法标记组合：

SYN/FIN

SYN/RST

SYN/FIN/PSH SYN/FIN/RST

SYN/FIN/RST/PSH FIN( 此 FIN 无 ACK)

“NULL”



# iptables

## 8. TCP 标志

### TCP 标志控制

控制 TCP 的标志位可以使用 --tcp-flags 选项  
主要有两个参数：

- 1) SYN,ACK,FIN,RST,URG,PSH 组合
- 2) 标志位为 1



# iptables

## 8. TCP 标志

1) 检查 SYN,ACK,FIN 的标志 . 但是数据包中必须设置了 SYN 的时候才是匹配的

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

2) 检查所有的标记  
(SYN,ACK,FIN,RST,URG,PSH). 但是必须设置了  
SYN 和 ACK 的才是匹配的

```
#iptables -A INPUT -p tcp --tcp-flags ALL  
SYN,ACK
```



# iptables

## 8. TCP 标志

3) 检查 SYN/RST/ACK 且 SYN 必须置位

```
#iptables -A INPUT -p tcp --syn
```



# iptables

## 9. LOG

在 -j LOG 可以将匹配规则记录至日志中。日志级别有：

debug ( 调试 7)

info( 信息 6)

notice( 需要处理但不是错误 5)

warning( 警告 4)(Linux 系统默认)



# iptables

## 9. LOG

在 -j LOG 可以将匹配规则记录至日志中。日志级别有：

err( 错误 3)

crit( 重要警告 2)

alert( 重要错误 , 需要立即更正 1)

emerg( 紧急情况 0)

由 debug->emerg 也可以由数字来代表 7->0

# iptables

## 9. LOG

示例：

```
#iptables -A INPUT -p icmp -m limit --limit 3/m --limit-burst 3 -j LOG --log-level 3 --log-prefix "INPUT packet died"
```



# iptables

## 10. NAT( 网络地址翻译 )

### 重复使用 IP 地址技术

NAT 主要提供了将一个内部网络地址映射到 INTERNET 地址的一个标准的方法。 NAT 可以将每个 LAN 中的节点转换成一个 INTERNET 地址。反之亦如此。把某个 IP 地址隐藏起来不会被外面发现。



# iptables

## 10. NAT( 网络地址翻译 )

NAT 用途：

1. 只需要一个公网 IP 地址就可以让 LAN 中的所有节点连接到 INTERNET
2. 实现透明代理等 .....



# iptables

## 10. NAT( 网络地址翻译 )

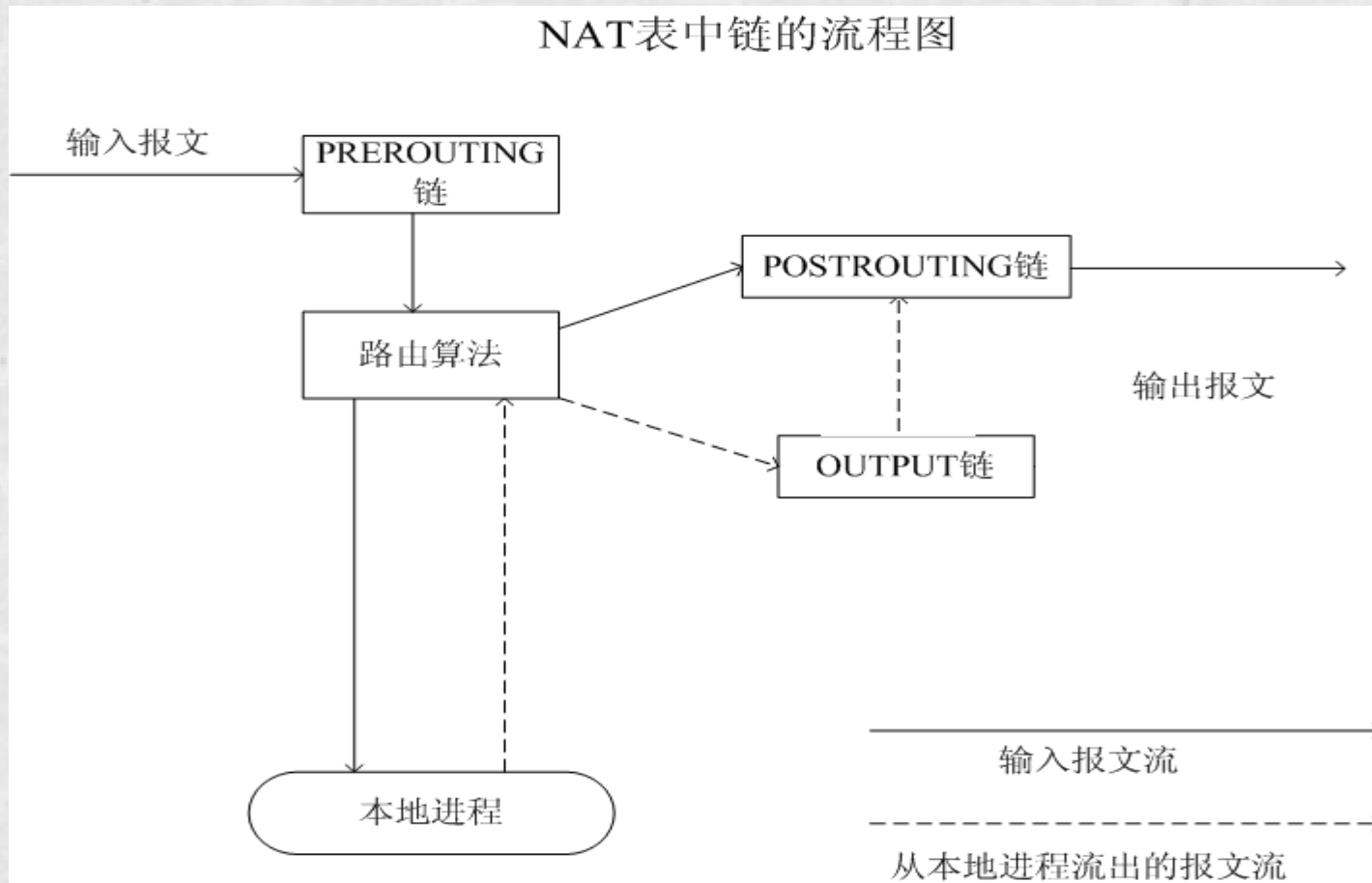
1. 源 NAT(SNAT):Source NAT 是指 : 修改包中源地址 (POSTROUTING)

2. 目的 NAT(DNAT):Destination NAT 是指 : 修改包中目的地址 (PREROUTING)。对于端口转发, 负载均衡, 透明代理都是 DNAT.



# iptables

## 10. NAT( 网络地址翻译 ) SNAT 与 DNAT 工作流程图





# iptables

## 10. NAT( 网络地址翻译 )

1. 在 nat 表中的 POSTROUTING 链中添加一条规则 . 将数据包的报头的 IP 地址改为 202.106.107.109 这个 IP, 并且将数据包的源端口改为 1024-65535 中任何一个可用的端口。

```
#iptables -t nat -A POSTROUTING -p tcp  
-o eth0  
-j SNAT --to 202.106.148.1:1024-65535
```

# iptables

## 10. NAT( 网络地址翻译 )

### 2. 简易 NAT

```
#!/bin/bash
echo -n "Starting Nat....."
/sbin/iptables -F
/sbin/iptables -P FORWARD ACCEPT
/sbin/iptables -t nat -A POSTROUTING -o
ppp+ -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward
echo "OK!!!!"
```



# iptables

## 10. NAT( 网络地址翻译 )

### 3. 透明代理

```
#!/bin/bash
```

```
/sbin/iptables -F -t filter
```

```
/sbin/iptables -F -t nat
```

```
/sbin/iptables -P INPUT ACCEPT
```

```
/sbin/iptables -P OUTPUT ACCEPT
```

```
/sbin/iptables -P FORWARD ACCEPT
```

```
/sbin/iptables -t nat -P PREROUTING ACCEPT
```



# iptables

## 10. NAT( 网络地址翻译 )

### 3. 透明代理

```
/sbin/iptables -t nat -P POSTROUTING  
ACCEPT
```

```
/sbin/iptables -t nat -P OUTPUT ACCEPT
```

```
/sbin/iptables -A INPUT -i lo -j ACCEPT
```

```
/sbin/iptables -A INPUT -i eth1 -j ACCEPT
```



# iptables

10. NAT( 网络地址翻译 )

3. 透明代理

```
/sbin/iptables -t nat -A POSTROUTING -s  
192.168.0.0/24 -j MASQUERADE
```

```
/sbin/iptables -t nat -A PREROUTING -p tcp  
-S  
192.168.0.0/24 --dport 80 -j DNAT --to  
192.168.0.1:3128
```

# iptables

## 10. NAT( 网络地址翻译 )

### 4. httpd 转发

```
iptables -t nat -A PREROUTING -p tcp  
--dport 80 -j DNAT --to-destination  
192.168.2.41:80
```

```
iptables -t nat -A POSTROUTING -s  
192.168.10/24 -d 192.168.2.41 -p tcp  
--dport 80 -j SNAT --to-source 192.168.2.39  
echo 1 > /proc/sys/net/ipv4/ip_forward
```

# iptables

## 状态控制

```
iptables -A INPUT -s 192.168.2.31 -d  
192.168.2.39 -p tcp -m state --state  
NEW,ESTABLISHED -m tcp --dport 22 -j  
ACCEPT
```

```
iptables -A OUTPUT -s 192.168.2.39 -d  
192.168.2.31 -p tcp --sport 22 -j ACCEPT
```

# iptables

控制并发链接

1) 添加内核模块

```
#modprobe ipt_connlimit
```

2) 允许 IP 的最大连接数为 30

```
#iptables -A INPUT -p tcp --dport 80 -m  
connlimit --connlimit-above 30 -j DROP
```





# iptables

## 控制并发链接

3) 限制连往本机的 web 服务，1 个 C 段的 IP 的并发连接不超过 100 个，超过的被拒绝：

```
#iptables -I INPUT -p tcp --dport 80 -m  
iplimit --iplimit-above 100 \  
--iplimit-mask 24 -j REJECT
```

4) 最多同时允许两个 IP 链接到 22 端口

```
#iptables -A INPUT -p tcp --dport 22 -m  
connlimit --connlimit-above 2 -j REJECT
```

# iptables

## FTP 主动模式

```
#iptables -A INPUT -m state --state  
NEW,RELATED,ESTABLISHED -j ACCEPT
```

## FTP 被动模式

### (1) 增加模块

```
#modprobe ip_conntrack  
#modprobe ip_conntrack_ftp  
#modprobe ip_nat_ftp
```



# iptables

FTP 被动模式

(2) 设定 vsftp 的端口

pasv\_min\_port=2222

pasv\_max\_port=2225



# iptables

FTP 被动模式

(3) 开启 iptables

```
#iptables -A INPUT -m state --state  
RELATED,ESTABLISHED -j ACCEPT
```

```
#iptables -A INPUT -p tcp -m state --state  
NEW -m tcp --dport 21 -j ACCEPT
```

```
#iptables -A INPUT -p tcp --dport  
2222:2225 -j ACCEPT
```



# iptables

FTP 被动模式

(4) 开启 ping

```
#iptables -A INPUT -s 192.168.38.100 -p  
icmp -m icmp --icmp-type 0 -j ACCEPT
```

```
#iptables -A INPUT -s 192.168.38.100  
-p icmp -m icmp --icmp-type 8 -j ACCEPT
```

```
#iptables -A INPUT -p icmp -m icmp  
--icmp-type 8 -j DROP
```

```
#iptables -A INPUT -p icmp -m icmp  
--icmp-type 0 -j DROP
```

# iptables

保存 iptables 规则

```
#iptables-save
```

保存目录及文件名称

```
#cat /etc/sysconfig/iptables
```

恢复 iptables 规则

```
#iptables-restore
```

