

# 两小时玩转 iptables

摘自 Chinaunix 讲座  
cu.platinum@gmail.com  
2006.03.18

## 主题大纲

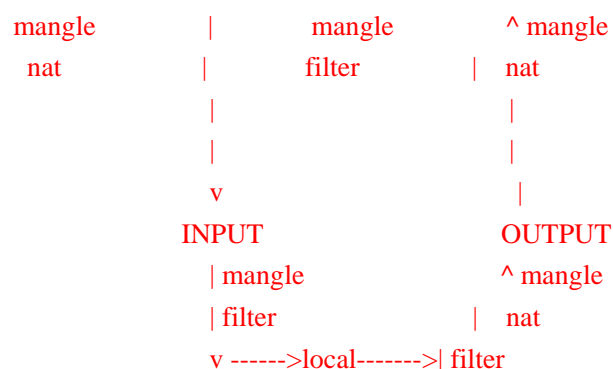
1. 概述
2. 框架图
3. 语法
4. 实例分析
5. 网管策略
6. FAQ
7. 实战

## 1. 概述

2.4.x、2.6.x 内核  
netfilter/iptables

### 2.1 框架图

-->PREROUTING-->[ROUTE]-->FORWARD-->POSTROUTING-->



### 2.2 链和表

#### ● 表

filter: 顾名思义，用于过滤的时候  
nat: 顾名思义，用于做 NAT 的时候  
NAT: Network Address Translator

#### ● 链

INPUT: 位于 filter 表，匹配目的 IP 是本机的数据包  
FORWARD: 位于 filter 表，匹配穿过本机的数据包，  
PREROUTING: 位于 nat 表，用于修改目的地址（DNAT）  
POSTROUTING: 位于 nat 表，用于修改源地址（SNAT）

### 3.1 iptables 语法概述

- **iptables [-t 要操作的表]**
  - <操作命令>
  - [要操作的链]
  - [规则号码]
  - [匹配条件]
  - [-j 匹配到以后的动作]

### 3.2 命令概述

- 操作命令 (-A、-I、-D、-R、-P、-F)
- 查看命令 (-[vnx]L)

#### 3.2.1 -A

##### -A <链名>

APPEND, 追加一条规则 (放到最后)

例如:

```
iptables -t filter -A INPUT -j DROP
```

在 filter 表的 INPUT 链里追加一条规则 (作为最后一条规则)  
匹配所有访问本机 IP 的数据包, 匹配到的丢弃

#### 3.2.2 -I

##### -I <链名> [规则号码]

INSERT, 插入一条规则

例如:

```
iptables -I INPUT -j DROP
```

在 filter 表的 INPUT 链里插入一条规则 (插入成第 1 条)

```
iptables -I INPUT 3 -j DROP
```

在 filter 表的 INPUT 链里插入一条规则 (插入成第 3 条)

注意: 1、-t filter 可不写, 不写则自动默认是 filter 表  
2、-I 链名 [规则号码], 如果不写规则号码, 则默认是 1  
3、确保规则号码  $\leq$  (已有规则数 + 1), 否则报错

#### 3.2.3 -D

##### -D <链名> <规则号码 | 具体规则内容>

DELETE, 删除一条规则

例如:

```
iptables -D INPUT 3 (按号码匹配)
```

删除 filter 表 INPUT 链中的第三条规则 (不管它的内容是什么)

```
iptables -D INPUT -s 192.168.0.1 -j DROP (按内容匹配)
```

删除 filter 表 INPUT 链中内容为“-s 192.168.0.1 -j DROP”的规则  
(不管其位置在哪里)

注意:

- 1、若规则列表中有多条相同的规则时, 按内容匹配只删除序号最小的一条
- 2、按号码匹配删除时, 确保规则号码  $\leq$  已有规则数, 否则报错
- 3、按内容匹配删除时, 确保规则存在, 否则报错

### 3.2.4 -R

**-R <链名> <规则号码> <具体规则内容>**

**REPLACE, 替换一条规则**

例如:

```
iptables -R INPUT 3 -j ACCEPT
```

将原来编号为 3 的规则内容替换为“-j ACCEPT”

注意:

确保规则号码  $\leq$  已有规则数, 否则报错

### 3.2.5 -P

**-P <链名> <动作>**

**POLICY, 设置某个链的默认规则**

例如:

```
iptables -P INPUT DROP
```

设置 filter 表 INPUT 链的默认规则是 DROP

注意:

当数据包没有被规则列表里的任何规则匹配到时, 按此默认规则处理

### 3.2.6 -F

**-F [链名]**

**FLUSH, 清空规则**

例如:

```
iptables -F INPUT
```

清空 filter 表 INPUT 链中的所有规则

```
iptables -t nat -F PREROUTING
```

清空 nat 表 PREROUTING 链中的所有规则

注意:

- 1、-F 仅仅是清空链中规则, 并不影响 -P 设置的默认规则
- 2、-P 设置了 DROP 后, 使用 -F 一定要小心!!!
- 3、如果不写链名, 默认清空某表里所有链里的所有规则

### 3.2.7 -[vxnl]

#### -L [链名]

**LIST**, 列出规则

v: 显示详细信息, 包括每条规则的匹配包数量和匹配字节数

x: 在 v 的基础上, 禁止自动单位换算 (K、M)

n: 只显示 IP 地址和端口号码, 不显示域名和服务名称

例如:

```
iptables -L
```

粗略列出 filter 表所有链及所有规则

```
iptables -t nat -vnL
```

用详细方式列出 nat 表所有链的所有规则, 只显示 IP 地址和端口号

```
iptables -t nat -vxnl PREROUTING
```

用详细方式列出 nat 表 PREROUTING 链的所有规则以及详细数字, 不反解

### 3.3 匹配条件

- 流入、流出接口 (-i、-o)
- 来源、目的地址 (-s、-d)
- 协议类型 (-p)
- 来源、目的端口 (--sport、--dport)

#### 3.3.1 按网络接口匹配

##### -i <匹配数据进入的网络接口>

例如:

```
-i eth0
```

匹配是否从网络接口 eth0 进来

```
-i ppp0
```

匹配是否从网络接口 ppp0 进来

##### -o 匹配数据流出的网络接口

例如:

```
-o eth0
```

```
-o ppp0
```

#### 3.3.2 按来源目的地址匹配

##### -s <匹配来源地址>

可以是 IP、NET、DOMAIN, 也可空 (任何地址)

例如:

```
-s 192.168.0.1 匹配来自 192.168.0.1 的数据包
```

```
-s 192.168.1.0/24 匹配来自 192.168.1.0/24 网络的数据包
```

```
-s 192.168.0.0/16 匹配来自 192.168.0.0/16 网络的数据包
```

**-d <匹配目的地址>**

可以是 IP、NET、DOMAIN，也可以空

例如：

```
-d 202.106.0.20    匹配去往 202.106.0.20 的数据包
-d 202.106.0.0/16 匹配去往 202.106.0.0/16 网络的数据包
-d www.abc.com     匹配去往域名 www.abc.com 的数据包
```

**3.3.3 按协议类型匹配****-p <匹配协议类型>**

可以是 TCP、UDP、ICMP 等，也可为空

例如：

```
-p tcp
-p udp
-p icmp --icmp-type 类型
ping: type 8      pong: type 0
```

**3.3.4 按来源目的端口匹配****--sport <匹配源端口>**

可以是个别端口，可以是端口范围

例如：

```
--sport 1000      匹配源端口是 1000 的数据包
--sport 1000:3000 匹配源端口是 1000-3000 的数据包（含 1000、3000）
--sport :3000     匹配源端口是 3000 以下的数据包（含 3000）
--sport 1000:     匹配源端口是 1000 以上的数据包（含 1000）
```

**--dport <匹配目的端口>**

可以是个别端口，可以是端口范围

例如：

```
--dport 80        匹配源端口是 80 的数据包
--dport 6000:8000 匹配源端口是 6000-8000 的数据包（含 6000、8000）
--dport :3000     匹配源端口是 3000 以下的数据包（含 3000）
--dport 1000:     匹配源端口是 1000 以上的数据包（含 1000）
```

注意：--sport 和 --dport 必须配合 -p 参数使用

**3.3.5 匹配应用举例****1、端口匹配**

```
-p udp --dport 53
```

匹配网络中目的地址是 53 的 UDP 协议数据包

**2、地址匹配**

```
-s 10.1.0.0/24 -d 172.17.0.0/16
```

匹配来自 10.1.0.0/24 去往 172.17.0.0/16 的所有数据包

**3、端口和地址联合匹配**

```
-s 192.168.0.1 -d www.abc.com -p tcp --dport 80
```

匹配来自 192.168.0.1，去往 www.abc.com 的 80 端口的 TCP 协议数据包

注意:

- 1、--sport、--dport 必须联合 -p 使用, 必须指明协议类型是什么
- 2、条件写的越多, 匹配越细致, 匹配范围越小

### 3.4 动作 (处理方式)

- ACCEPT
- DROP
- SNAT
- DNAT
- MASQUERADE

#### 3.4.1 -j ACCEPT

##### -j ACCEPT

通过, 允许数据包通过本链而不拦截它  
类似 Cisco 中 ACL 里面的 permit

例如:

```
iptables -A INPUT -j ACCEPT
```

允许所有访问本机 IP 的数据包通过

#### 3.4.2 -j DROP

##### -j DROP

丢弃, 阻止数据包通过本链而丢弃它  
类似 Cisco 中 ACL 里的 deny

例如:

```
iptables -A FORWARD -s 192.168.80.39 -j DROP
```

阻止来源地址为 192.168.80.39 的数据包通过本机

#### 3.4.3 -j SNAT

##### -j SNAT --to IP[-IP][:端口-端口] (nat 表的 POSTROUTING 链)

源地址转换, SNAT 支持转换为单 IP, 也支持转换到 IP 地址池  
(一组连续的 IP 地址)

例如:

```
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 \
```

```
-j SNAT --to 1.1.1.1
```

将内网 192.168.0.0/24 的原地址修改为 1.1.1.1, 用于 NAT

```
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 \
```

```
-j SNAT --to 1.1.1.1-1.1.1.10
```

同上, 只不过修改成一个地址池里的 IP

#### 3.4.4 -j DNAT

##### -j DNAT --to IP[-IP][:端口-端口] (nat 表的 PREROUTING 链)

目的地址转换, DNAT 支持转换为单 IP, 也支持转换到 IP 地址池

（一组连续的 IP 地址）

例如：

```
iptables -t nat -A PREROUTING -i ppp0 -p tcp --dport 80 \
-j DNAT --to 192.168.0.1
```

把从 ppp0 进来的要访问 TCP/80 的数据包目的地址改为 192.168.0.1

```
iptables -t nat -A PREROUTING -i ppp0 -p tcp --dport 81 \
-j DNAT --to 192.168.0.2:80
```

```
iptables -t nat -A PREROUTING -i ppp0 -p tcp --dport 80 \
-j DNAT --to 192.168.0.1-192.168.0.10
```

### 3.4.5 -j MASQUERADE

#### -j MASQUERADE

动态源地址转换（动态 IP 的情况下使用）

例如：

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

将源地址是 192.168.0.0/24 的数据包进行地址伪装

## 3.5 附加模块

- 按包状态匹配 (state)
- 按来源 MAC 匹配 (mac)
- 按包速率匹配 (limit)
- 多端口匹配 (multiport)

### 3.5.1 state

#### -m state --state 状态

状态：NEW、RELATED、ESTABLISHED、INVALID

NEW：有别于 tcp 的 syn

ESTABLISHED：连接态

RELATED：衍生态，与 conntrack 关联（FTP）

INVALID：不能被识别属于哪个连接或没有任何状态

例如：

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

### 3.5.2 mac

#### -m mac --mac-source MAC

匹配某个 MAC 地址

例如：

```
iptables -A FORWARD -m --mac-source xx:xx:xx:xx:xx:xx \
-j DROP
```

阻断来自某 MAC 地址的数据包，通过本机

注意：

MAC 地址不过路由，不要试图去匹配路由后面的某个 MAC 地址

### 3.5.3 limit

**-m limit --limit 匹配速率 [--burst 缓冲数量]**

用一定速率去匹配数据包

例如：

```
iptables -A FORWARD -d 192.168.0.1 -m limit --limit 50/s \  
-j ACCEPT  
iptables -A FORWARD -d 192.168.0.1 -j DROP
```

注意：

limit 仅仅是用一定的速率去匹配数据包，并非“限制”

### 3.5.4 multiport

**-m multiport <--sports|--dports|--ports> 端口 1[,端口 2,...,端口 n]**

一次性匹配多个端口，可以区分源端口，目的端口或不指定端口

例如：

```
iptables -A INPUT -p tcp -m multiports --ports \  
21,22,25,80,110 -j ACCEPT
```

注意：

必须与 -p 参数一起使用

## 4. 实例分析

- 单服务器的防护
- 如何做网关
- 如何限制内网用户
- 内网如何做对外服务器
- 连接追踪模块

### 4.1 单服务器的防护

- 弄清对外服务对象
- 书写规则

网络接口 lo 的处理

状态监测的处理

协议 + 端口的处理

实例：一个普通的 web 服务器

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

```
iptables -A INPUT -p tcp -m multiport 22,80 -j ACCEPT
```

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

```
iptables -P INPUT DROP
```

注意：确保规则循序正确，弄清逻辑关系，学会时刻使用 -vnL



## 4.2 如何做网关

- 弄清网络拓扑
- 本机上网
- 设置 **nat**
  - 启用路由转发
  - 地址伪装 **SNAT/MASQUERADE**

实例：ADSL 拨号上网的拓扑

```
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ppp0 \
-j MASQUERADE
```

## 4.3 如何限制内网用户

- 过滤位置 **filter** 表 **FORWARD** 链
- 匹配条件 **-s -d -p --s/dport**
- 处理动作 **ACCEPT DROP**

实例：

```
iptables -A FORWARD -s 192.168.0.3 -j DROP
iptables -A FORWARD -m mac --mac-source 11:22:33:44:55:66 \
-j DROP
iptables -A FORWARD -d bbs.chinaunix.net -j DROP
```

## 4.4 内网如何做对外服务器

- 服务协议（**TCP/UDP**）
- 对外服务端口
- 内部服务器私网 **IP**
- 内部真正服务端口

实例：

```
iptables -t nat -A PREROUTING -i ppp0 -p tcp --dport 80 \
-j DNAT --to 192.168.1.1
iptables -t nat -A PREROUTING -i ppp0 -p tcp --dport 81 \
-j DNAT --to 192.168.1.2:80
```

## 4.5 连接追踪模块

- 为什么要使用连接追踪模块
  - FTP** 协议的传输原理
  - 传统防火墙的做法
- 如何使用

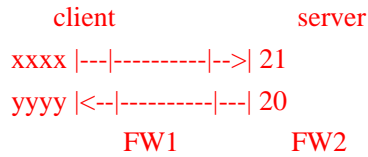
### 4.5.1 FTP 协议传输原理

- 使用端口
  - command port**
  - data port**
- 传输模式

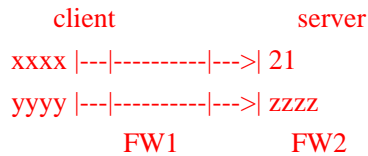
主动模式 (ACTIVE)

被动模式 (PASSIVE)

主动模式



被动模式



#### 4.5.2 传统防火墙的做法

- 只使用主动模式，打开 TCP/20
- 防火墙打开高范围端口
- 配置 FTP 服务，减小被动模式端口范围

#### 4.5.3 如何使用连接追踪模块

```
modprobe ipt_conntrack_ftp
modprobe ipt_nat_ftp
iptables -A INPUT -p tcp --dport 21 -j ACCEPT
iptables -A INPUT -m state --state \
    RELATED,ESTABLISHED -j ACCEPT
iptables -P INPUT DROP
```

### 5. 网管策略

- 怕什么
- 能做什么
- 让什么 vs 不让什么
- 三大“纪律”五项“注意”
- 其他注意事项

#### 5.1 必加项

```
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/tcp_syncookies
echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
modprobe ip_conntrack_ftp
modprobe ip_nat_ftp
```

#### 5.2 可选方案

堵:

```
iptables -A FORWARD -p tcp --dport xxx -j DROP
iptables -A FORWARD -p tcp --dport yyy:zzz -j DROP
```

通:

```
iptables -A FORWARD -p tcp --dport xxx -j ACCEPT
```

```
iptables -A FORWARD -p tcp --dport yyy:zzz -j ACCEPT
```

```
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

```
iptables -P FORWARD DROP
```

### 5.3 三大“纪律”五项“注意”

#### ● 三大“纪律”——专表专用

filter

nat

mangle

#### ● 五项“注意”——注意数据包的走向

PREROUTING

INPUT

FORWARD

OUTPUT

POSTROUTING

### 5.4 其他注意事项

#### ● 养成好的习惯

```
iptables -vnL
```

```
iptables -t nat -vnL
```

```
iptables-save
```

#### ● 注意逻辑顺序

```
iptables -A INPUT -p tcp --dport xxx -j ACCEPT
```

```
iptables -I INPUT -p tcp --dport yyy -j ACCEPT
```

#### ● 学会写简单的脚本

## 6. FAQ.1

Q: 我设置了 `iptables -A OUTPUT -d 202.xx.xx.xx -j DROP`

为何内网用户还是可以访问那个地址?

A: filter 表的 OUTPUT 链是本机访问外面的必经之路, 内网数据不经过该链

Q: 我添加了 `iptables -A FORWARD -d 202.xx.xx.xx -j DROP`

为何内网用户还是可以访问那个地址?

A: 检查整个规则是否存在逻辑错误, 看是否在 DROP 前有 ACCEPT

Q: `iptables -t nat -A POSTROUTING -i eth1 -o eth2 -j MASQUERADE`

这条语句为何报错?

A: POSTROUTING 链不支持“流入接口” -i 参数

同理, PREROUTING 链不支持“流出接口” -o 参数

## 6. FAQ.2

Q: 我应该怎么查看某个模块具体该如何使用?

A: `iptables -m 模块名 -h`

Q: 执行 `iptables -A FORWARD -m xxx -j yyy`

提示 `iptables: No chain/target/match by that name`

A: `/lib/modules/$(uname -r)/kernel/net/ipv4/netfilter` 目录中,

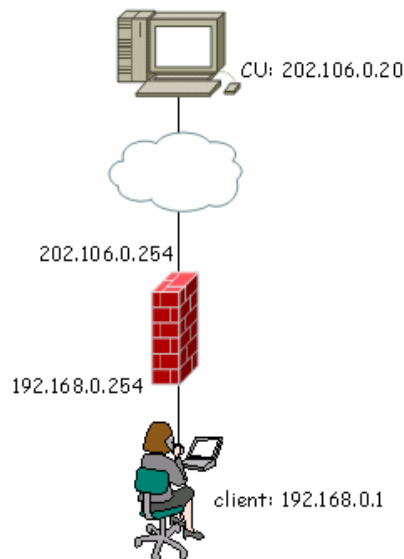
缺少与 `xxx` 模块有关的文件, 或缺少与 `yyy` 动作有关的文件

名字为 `ipt_xxx.o` (2.4 内核) 或 `ipt_yyy.ko` (2.6 内核)

Q: 脚本写好了, 内网上网没问题, FTP 访问不正常, 无法列出目录, 为什么?

A: 缺少 `ip_nat_ftp` 这个模块, `modprobe ip_nat_ftp`

## 7. 实战 1



## 7. 实战.2

