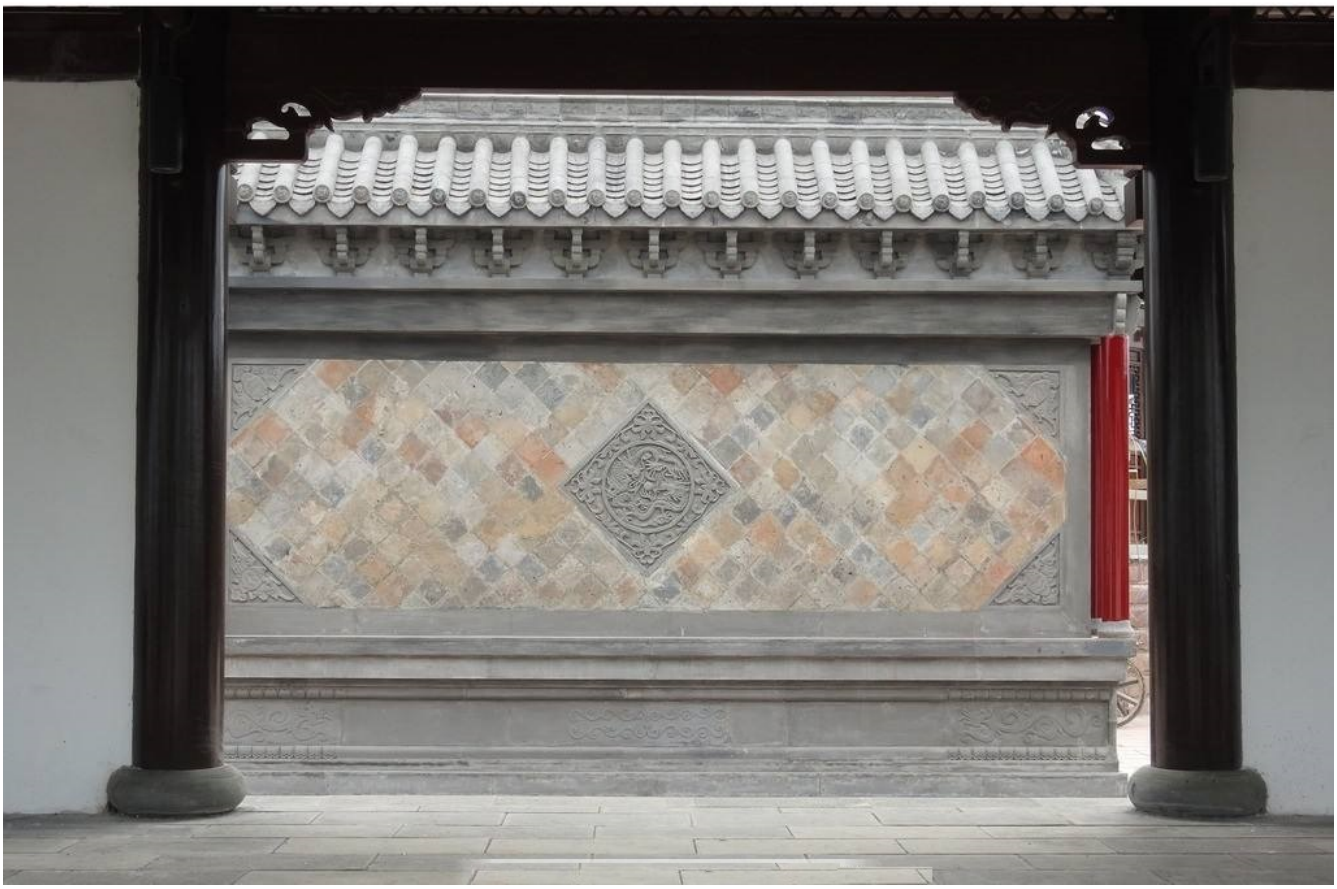
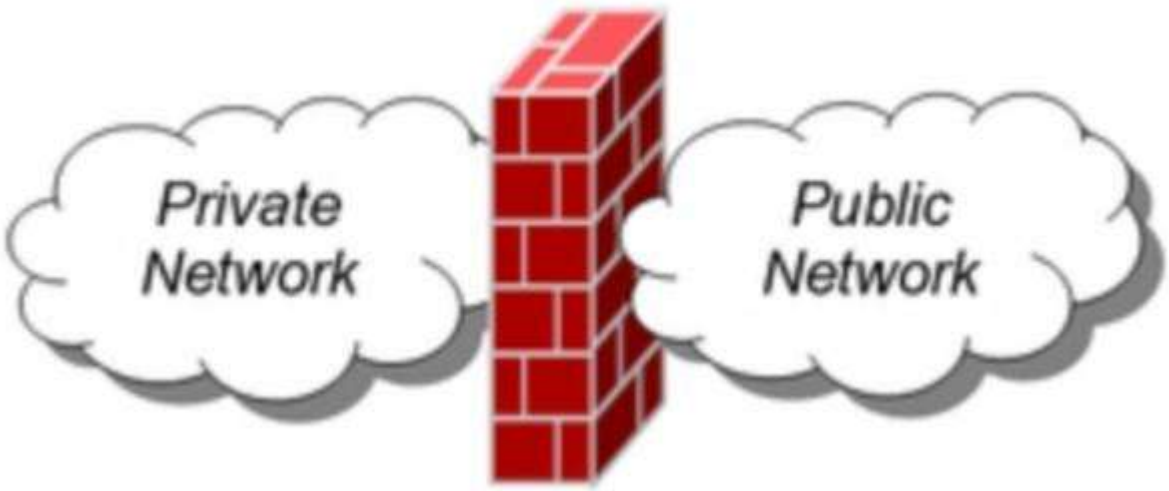
防火墙(Firewall)

防火墙作用



在计算机领域，防火墙是用于保护信息安全的设备，其会依照用户定义的规则，允许或限制数据的传输。



- 用于保护内网安全的一种设备
- 依据规则进行防护
- 用户定义规则
- 允许或拒绝外部用户访问

防火墙分类

- 逻辑上划分，防火墙可以大体分为主机防火墙和网络防火墙

主机防火墙：针对于单个主机进行防护

网络防火墙：针对网络进行防护，处于网络边缘，防火墙背后是本地局域网

网络防火墙主外(服务集体)，主机防火墙主内(服务个人)

- 物理上划分，防火墙可分为硬件防火墙和软件防火墙

硬件防火墙：在硬件级别实现防火墙功能，另一部分基于软件实现，其性能高，硬件成本高

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，其性能相较于硬件防火墙低，成本较低，对于Linux系统已自带，直接使用即可

防火墙性能

- 吞吐量

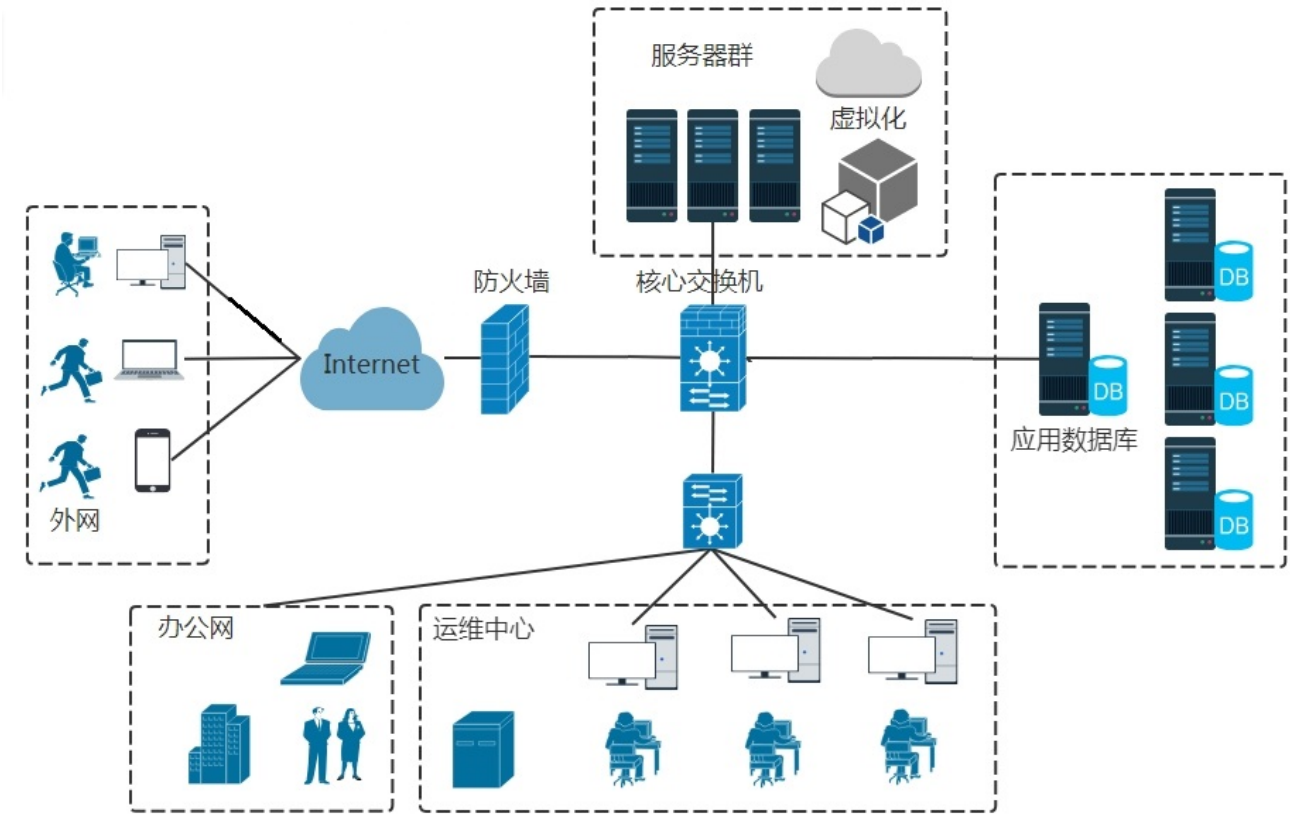
- 并发连接
- 新建连接
- 时延
- 抖动

硬件防火墙

硬件防火墙定义

硬件防火墙是指把具备配置数据包通过规则的软件嵌入硬件设备中，为网络提供安全防护的硬件设备。多见于网络边缘。

硬件防火墙作用



硬件防火墙品牌

- Juniper

- cisco 思科ASA

- 华为

- 天融信

软件防火墙

软件防火墙是单独使用具备配置数据包通过规则的软件来实现数据包过滤。多见于单主机系统或个人计算机。

扩展：Web应用防火墙(WAF)

Web应用防火墙是对web防护(网页保护)的安全防护设备(软件)，主要用于截获所有HTTP数据或仅仅满足某些规则的会话。多见于云平台中。

硬件防火墙与软件防火墙比较

硬件防火墙有独立的硬件设备，运算效率较高，价格略高，可为计算机网络提供安全防护。

软件防火墙必须部署在主机系统之上，相较于硬件防火墙运算效率低，在一定程度上会影响到主机系统性能，一般用于单机系统或个人计算机中，不直接用于计算机网络中。

iptables

iptables是什么？

- iptables不是防火墙，是防火墙用户代理
- 用于把用户的安全设置添加到“安全框架”中
- “安全框架”是防火墙
- “安全框架”的名称为netfilter
- netfilter位于内核空间中，是Linux操作系统核心层内部的一个数据包处理模块
- iptables是用于在用户空间对内核空间的netfilter进行操作的命令行工具

netfilter/iptables功能

netfilter/iptables可简称为iptables，为Linux平台下的包过滤防火墙，是开源的，内核自带的，可以代替成本较高的企业级硬件防火墙，能够实现如下功能：

- 数据包过滤，即防火墙
- 数据包重定向，即转发
- 网络地址转换，即可NAT

注：

平常我们使用iptables并不是防火墙的“服务”，而服务是由内核提供的。

iptables概念

iptables工作依据------规则(rules)

iptables是按照规则(rules)来办事的，而规则就是运维人员所定义的条件；规则一般定义为“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数据包过滤表中，这些规则分别指定了源地址、目的地址，传输协议(TCP、UDP、ICMP)和服务类型(HTTP、FTP)等。

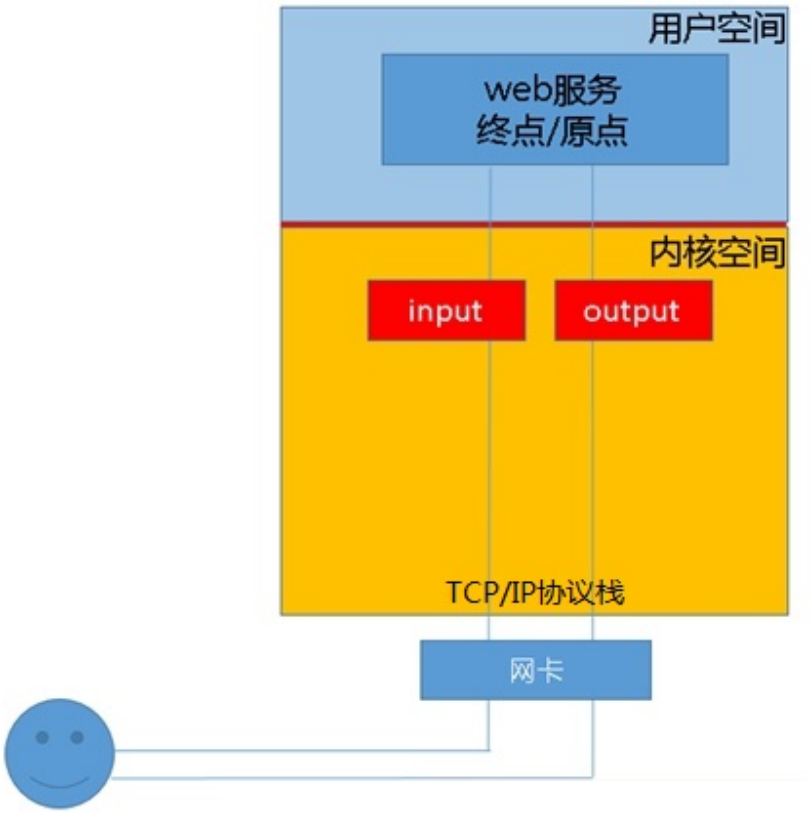
当数据包与规则匹配时，iptables就根据规则所定义的方法来处理这些数据包，比如放行(ACCEPT)、拒绝(REJECT)、丢弃(DROP)等

> 配置防火墙主要工作就是对iptables规则进行添加、修改、删除等

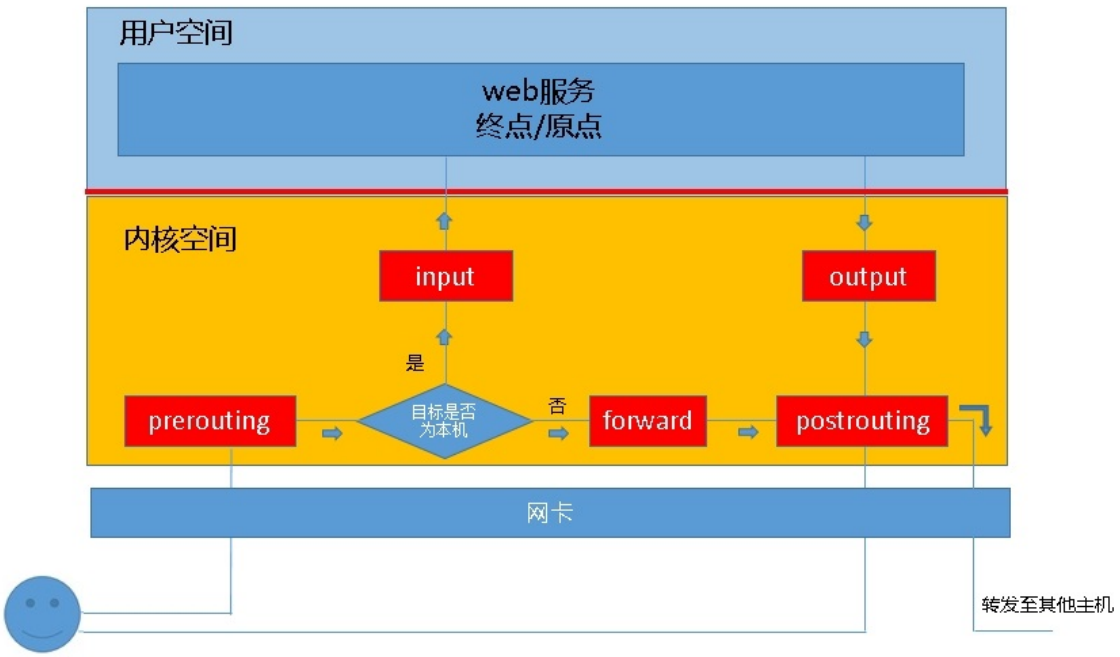
iptables中链的概念

举例说明：

当客户端访问服务器端的web服务时，客户端发送访问请求报文至网卡，而tcp/ip协议栈是属于内核的一部分，所以，客户端的请求报文会通过内核的TCP协议传输到用户空间的web服务，而客户端报文的目标地址为web服务器所监听的套接字(ip:port)上，当web服务器响应客户端请求时，web服务所回应的响应报文的目标地址为客户端地址，我们说过，netfilter才是真正的防火墙，属于内核的一部分，所以，我们要想让netfilter起到作用，我们就需要在内核中设置“关口”，所以进出的数据报文都要通过这些关口，经检查，符合放行条件的准允放行，符合阻拦条件的则被阻止，于是就出现了input和output关口，然而在iptables中我们把关口叫做“链”。

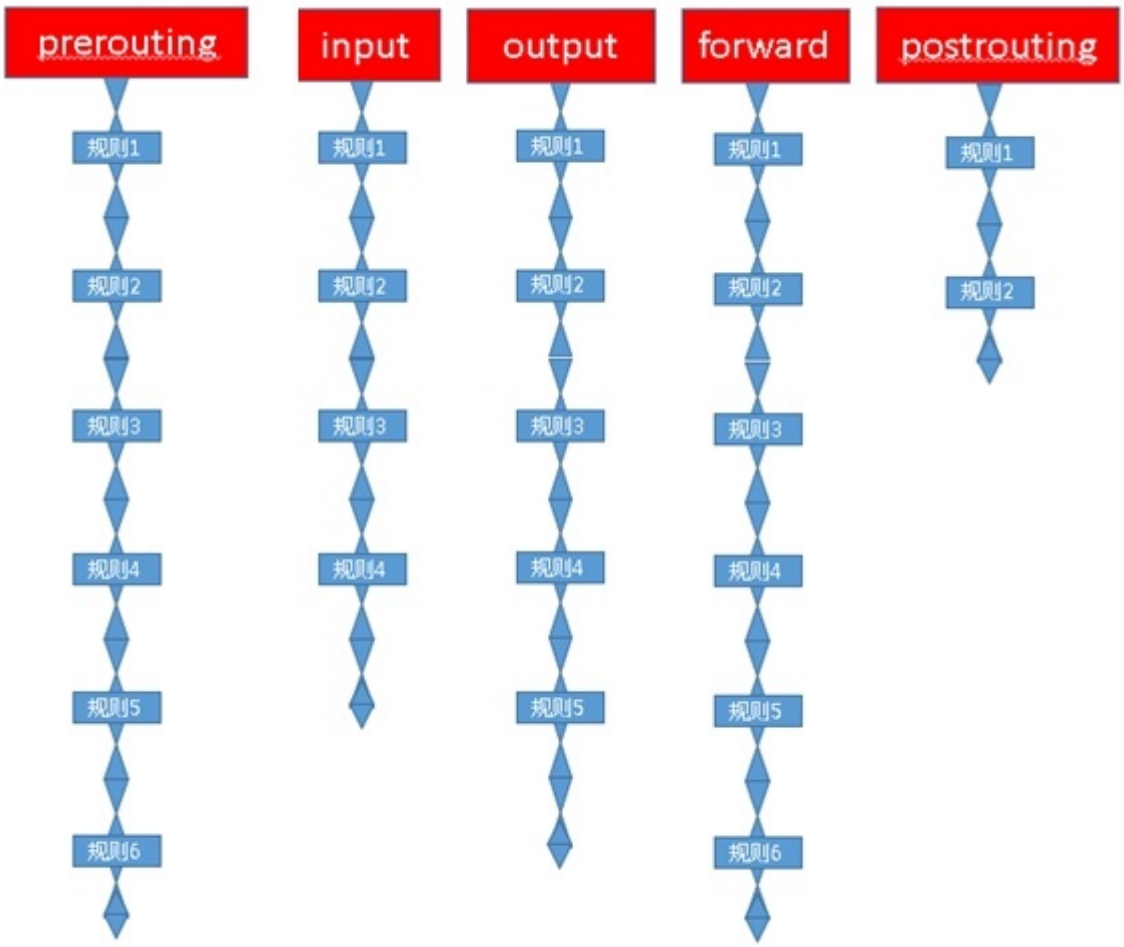


上面的举例中，如果客户端发到本机的报文中包含的服务器地址并不是本机，而是其他服务器，此时本机就应该能够进行转发，那么这个转发就是本机内核所支持的IP_FORWARD功能，此时我们的主机类似于路由器功能，所以我们会看到在iptables中，所谓的关口并只有上面所提到的input及output这两个，应该还有“路由前”，“转发”，“路由后”，它们所对应的英文名称分别为“PREROUTING”，“FORWARD”，“POSTROUTING”，这就是我们说到的5链。



通过上图可以看出，当我们在本地启动了防火墙功能时，数据报文需要经过以上关口，根据各报文情况，各报名经常的“链”可能不同，如果报文目标地址是本机，则会经常input链发往本机用户空间，如果报文目标不是本机，则会直接在内核空间中经常forward链和postrouting链转发出去。

有的时候我们也经常听到人们在称呼input为“规则链”，这又是怎么回事呢？我们知道，防火墙的作用在于对经过的数据报文进行规则匹配，然后执行对应的“动作”，所以数据包经过这些关口时，必须匹配这个关口规则，但是关口规则可能不止一条，可能会有很多，当我们把众多规则放在一个关口上时，所有的数据包经常都要进行匹配，那么就形成了一个要匹配的规则链条，因此我们也把“链”称作“规则链”。



- INPUT:处理入站数据包
- OUTPUT:处理出站数据包
- FORWARD:处理转发数据包(主要是将数据包转发至本机其它网卡)

  当数据报文经过本机时，网卡接收数据报文至缓冲区，内核读取报文ip首部，发现报文不是送到本机时（目的ip不是本机），由内核直接送到forward链做匹配，匹配之后若符合forward的规则，再经由postrouting送往下一跳或目的主机。
- PREROUTING:在进行路由选择前处理数据包，修改到达防火墙数据包的目的IP地址，用于判断目标主机

- POSTROUTING:在进行路由选择后处理数据包，修改要离开防火墙数据包的源IP地址，判断经由哪一接口送往下一跳

iptables中表的概念

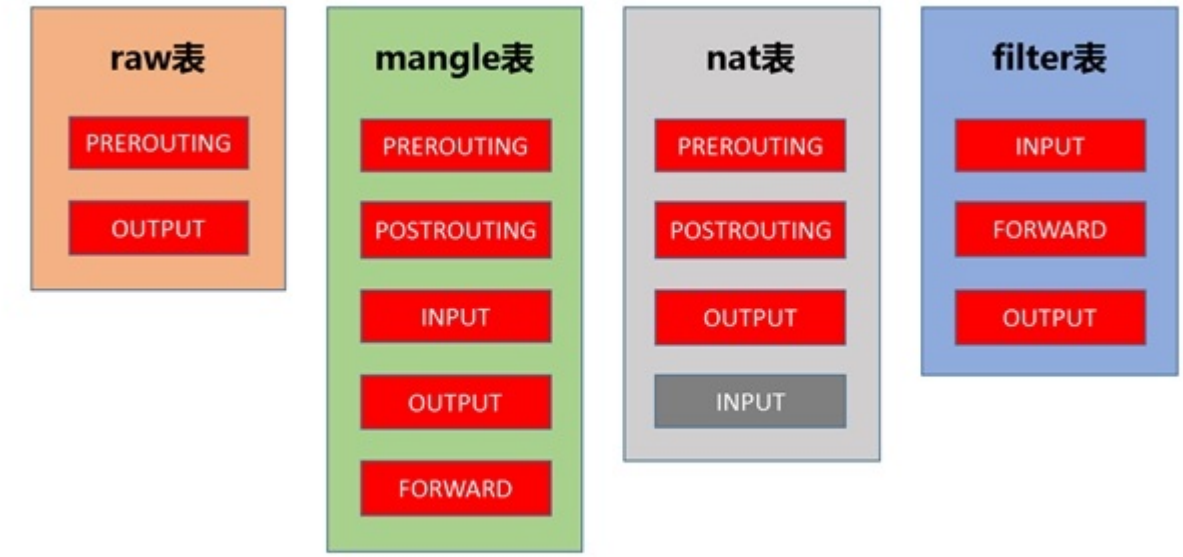
每个“规则链”上都设置了一串规则，这样的话，我们就可以把不同的“规则链”组合成能够完成某一特定功能集合分类，而这个集合分类我们就称为表，iptables中共有5张表，学习iptables需要搞明白每种表的作用。

- filter: 过滤功能，确定是否放行该数据包，属于真正防火墙，内核模块：iptables_filter
- nat: 网络地址转换功能，修改数据包中的源、目标IP地址或端口；内核模块：iptable_nat
- mangle: 对数据包进行重新封装功能，为数据包设置标记；内核模块：iptable_mangle
- raw: 确定是否对数据包进行跟踪；内核模块：iptables_raw
- security:是否定义强制访问控制规则；后加上的

iptables中表链之间的关系

我们在应用防火墙时是以表为操作入口的，只要在相应的表中的规则链上添加规则即可实现某一功能。那么我们就应该知道哪张表包括哪些规则链，然后在规则链上操作即可。

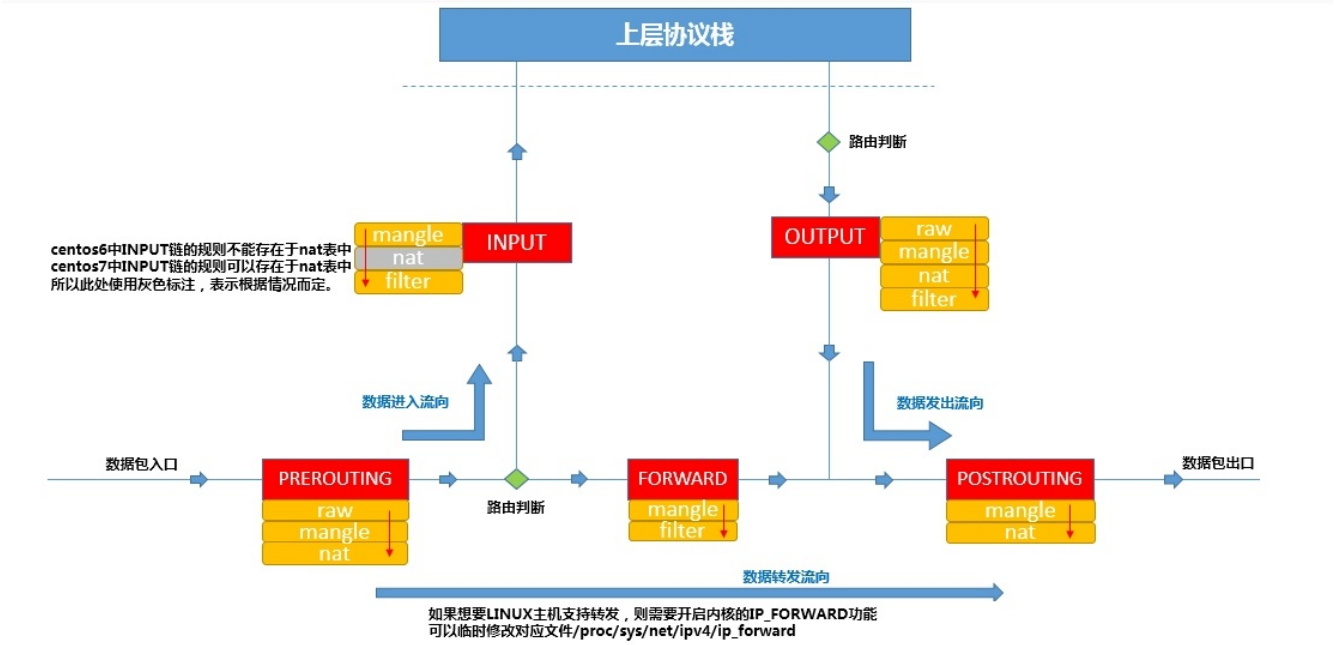
- filter表可以使用哪些链定义规则：INPUT,FORWARD,OUTPUT
- nat表中可以使用哪些链定义规则：PREROUTING,OUTPUT ,POSTROUTING,INPUT
- mangle 表中可以使用哪些链定义规则：PREROUTING,INPUT,FORWARD,OUTPUT,POSTROUTING
- raw表中可以使用哪些链定义规则：PREROUTING,OUTPUT



iptables中表的优先级

raw-mangle-nat-filter(由高至低)

数据包流经iptables流程



iptables规则匹配及动作

规则：根据指定的匹配条件来尝试匹配每个流经此处的数据包，匹配成功，则由规则指定的处理动作进行处理。

规则是由匹配条件和动作组成的，那么规则是什么呢？

举例说明：

两个同学，一个白头发，一个黑头发，同时进教室，而进教室的条件是只有黑头发可以进入，白头发拒绝进入，黑头发和白头发就是匹配条件，而可以进入和拒绝进入就是动作。

iptables规则匹配条件分类

基本匹配条件：

源地址，目标地址，源端口，目标端口等

基本匹配使用选项及功能

```
-p 指定规则协议，tcp udp icmp all
-s 指定数据包的源地址，ip hostname
-d 指定目的地址
-i 输入接口
-o 输出接口
! 取反
```

基本匹配的特点是：无需加载扩展模块，匹配规则生效

扩展匹配条件：

扩展匹配又分为显示匹配和隐式匹配。

扩展匹配的特点是：需要加载扩展模块，匹配规则方可生效。

隐式匹配的特点：使用-p选项指明协议时，无需再同时使用-m选项指明扩展模块以及不需要手动加载扩展模块；

显示匹配的特点：必须使用-m选项指明要调用的扩展模块的扩展机制以及需要手动加载扩展模块。

隐式匹配选项及功能

```
-p tcp
  --sport 匹配报文源端口；可以给出多个端口，但只能是连续的端口范围
  --dport 匹配报文目标端口；可以给出多个端口，但只能是连续的端口范围
  --tcp-flags mask comp 匹配报文中的tcp协议的标志位
-p udp
  --sport 匹配报文源端口；可以给出多个端口，但只能是连续的端口范围
  --dport 匹配报文目标端口；可以给出多个端口，但只能是连续的端口范围
--icmp-type
  0/0： echo reply 允许其他主机ping
  8/0：echo request 允许ping其他主机
```

显式匹配使用选项及功能

- multiport

  多端口

```
iptables -I INPUT -d 192.168.2.10 -p tcp -m multiport --dports 22,80 -j ACCEPT
#在INPUT链中开放本机tcp 22，tcp80端口

iptables -I OUTPUT -s 192.168.2.10 -p tcp -m multiport --sports 22,80 -j ACCEPT
#在OUTPUT链中开发源端口tcp 22，tcp80
```

- iprange

  多ip地址

```
iptables -A INPUT -d 192.168.2.10 -p tcp --dport 23 -m iprange --src-range
192.168.2.11-192.168.2.21 -j ACCEPT
iptables -A OUTPUT -s 192.168.2.10 -p tcp --sport 23 -m iprange --dst-range
192.168.2.11-192.168.2.21 -j ACCEPT
```

- time

  指定访问时间范围

```
iptables -A INPUT -d 192.168.2.10 -p tcp --dport 901 -m time --weekdays
Mon,Tus,Wed,Thu,Fri --timestart 08:00:00 --time-stop 18:00:00 -j ACCEPT

iptables -A OUTPUT -s 192.168.2.10 -p tcp --sport 901 -j ACCEPT
```

- string

  字符串，对报文中的应用层数据做字符串模式匹配检测(通过算法实现)。

```
--algo {bm|kmp}：字符匹配查找时使用算法
--string "STRING"： 要查找的字符串
--hex-string “HEX-STRING”: 要查找的字符，先编码成16进制格式
```

- connlimit

  连接限制，根据每个客户端IP作并发连接数量限制。

```
 --connlimit-upto n  连接数小于等于n时匹配
 --connlimit-above n 连接数大于n时匹配
```

- limit

  报文速率限制

- state

  追踪本机上的请求和响应之间的数据报文的状态。状态有五种：INVALID, ESTABLISHED, NEW, RELATED, UNTRACKED.

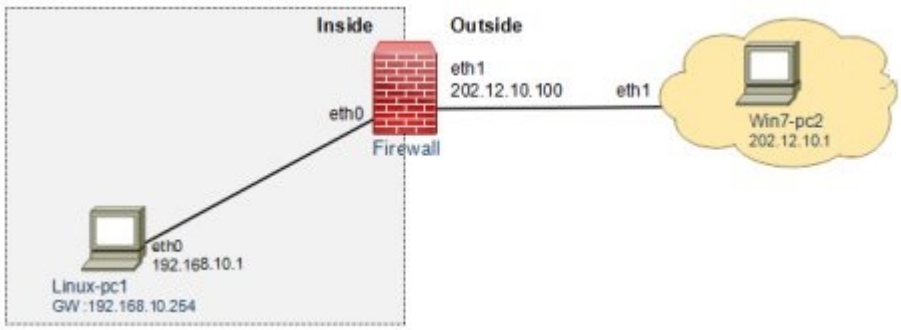
```
--state state
NEW 新连接请求
ESTABLISHED 已建立的连接
INVALID 无法识别的连接
RELATED 相关联的连接，当前连接是一个新请求，但附属于某个已存在的连接
UNTRACKED 未追踪的连接
```

```
1、对于进入的状态为ESTABLISHED都应该放行；

2、对于出去的状态为ESTABLISHED都应该放行；

3、严格检查进入的状态为NEW的连接；

4、所有状态为INVALIED都应该拒绝；
```

iptables规则中动作

iptables规则中的动作常称为target，也分为基本动作和扩展动作。

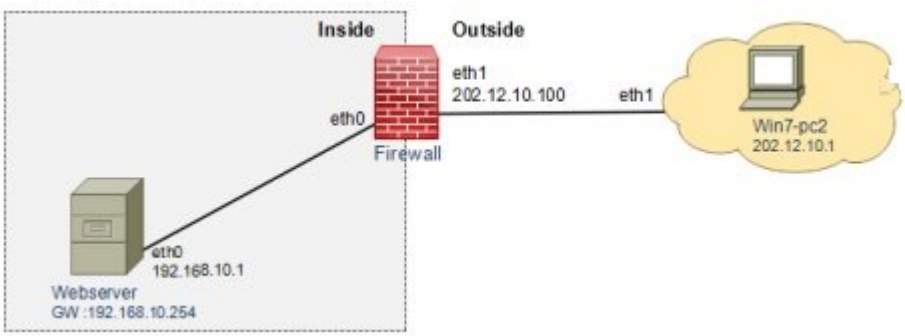
- ACCEPT:允许数据包通过
- DROP:直接丢弃数据包，不给任何回应信息
- REJECT:拒绝数据包通过，发送回应信息给客户端
- SNAT:源地址转换
    - 解释1：数据包从网卡发送出去的时候，把数据包中的源地址部分替换为指定的IP，接收方认为数据包的来源是被替换的那个IP主机，返回响应时，也以被替换的IP地址进行。
    - 解释2：修改数据包源地址，当内网数据包到达防火墙后，防火墙会使用外部地址替换掉数据包的源IP地址（目的IP地址不变），使网络内部主机能够与网络外部主机通信。




```
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -o eth1 -j SNAT --to-source
202.12.10.100
```

- MASQUERADE:伪装，类似于SNAT，适用于动态的、临时会变的ip地址上，例如：家庭使用的宽带。

  用发送数据的网卡上的IP来替换源IP，对于IP地址不固定场合使用。
- DNAT:目标地址转换
    - 解释1：数据包从网卡发出时，修改数据包中的目的IP，表现为你想访问A，但因网关做了DNAT，把所有访问A的数据包中的目的IP地址全部修改为B,实际最终访问的是B。
    - 解释2：改变数据包目的地址，当防火墙收到来自外网的数据包后，会将该数据包的目的IP地址进行替换（源IP地址不变），重新转发到内网的主机。



```
iptables -t nat -A PREROUTING -d 202.12.10.100 -p tcp --dport 80 -j DNAT --to-destination 192.168.10.1
```

- REDIRECT:在本机做端口映射
- LOG:在/var/log/message文件中记录日志信息，然后将数据包传递给下一条规则。

> 路由是按照目的地址进行路由选择的，因此，DNAT是在PREROUTING链上进行的，SNAT是在数据包发出的时候进行的，因此是在POSTROUTING链上进行的。

制定iptables规则策略

- 黑名单

  没有被拒绝的流量都可以通过，这种策略下管理员必须针对每一种新出现的攻击，制定新的规则，因此不推荐
- 白名单

  没有被允许的流量都要拒绝，这种策略比较保守，根据需要，主机主机逐渐开放，目前一般都采用白名单策略，推荐。

制定iptables规则思路

1. 选择一张表，此表决定了数据报文处理的方式
2. 选择一条链，此链决定了数据报文流经哪些位置
3. 选择合适的条件，此条件决定了对数据报文做何种条件匹配
4. 选择处理数据报文的动作，制定相应的防火墙规则

iptables基础语法结构

```
iptables [-t 表名] 管理选项 [链名] [条件匹配] [-j 目标动作或跳转]
```

> 不指定表名时，默认表示filter表，不指定链名时，默认表示该表内所有链，除非设置规则链的默认策略，否则需要指定匹配条件。

iptables规则组成

组成部分：四张表+五条链(Hook point)+规则

	table	command	chain	Parameter & Xmatch	target
iptables	-t filter nat mangle raw	-A -D -L -F -P -I -R -n	INPUT FORWARD OUTPUT PREROUTING POSROUTING	-p tcp -s -d --sport --dport --dports -m tcp -m state -m multiport	-j ACCEPT DROP REJECT DNAT SNAT

iptables链管理方法

```
-N, --new-chain chain：新建一个自定义的规则链；
-X, --delete-chain [chain]：删除用户自定义的引用计数为0的空链；
-F, --flush [chain]：清空指定的规则链上的规则；
-E, --rename-chain old-chain new-chain：重命名链；
-Z, --zero [chain [rulenum]]：置零计数器；
  注意：每个规则都有两个计数器
  packets：被本规则所匹配到的所有报文的个数；
  bytes：被本规则所匹配到的所有报文的大小之和；
-P, --policy chain target 制定链表的策略(ACCEPT|DROP|REJECT)
```

iptables规则管理

```
-A, --append chain rule-specification：追加新规则于指定链的尾部；
-I, --insert chain [rulenum] rule-specification：插入新规则于指定链的指定位置，默认为首部；
-R, --replace chain rulenum rule-specification：替换指定的规则为新的规则；
-D, --delete chain rulenum：根据规则编号删除规则；
-D, --delete chain rule-specification：根据规则本身删除规则；
```

iptables规则显示

```
    -L, --list [chain]：列出规则；
    -v, --verbose：详细信息；
    -vv 更详细的信息
    -n, --numeric：数字格式显示主机地址和端口号；
    -x, --exact：显示计数器的精确值，而非圆整后的数据；
    --line-numbers：列出规则时，显示其在链上的相应的编号；
    -S, --list-rules [chain]：显示指定链的所有规则；
```

iptables应用

- 环境 centos7

iptables-services安装

```
[root@localhost ~]# yum -y install iptables-services
```

> centos7系统中默认存在iptables命令，此命令仅为简单查询及操作命令，不包含配置文件，安装iptables-services后，将直接生成配置文件，便于配置保存。包含ipv4及ipv6。

- 设置服务开启

```
[root@localhost ~]# systemctl start iptables.service
```

- 设置开机自启动

```
[root@localhost ~]# systemctl enable iptables.services
```

- 查看配置文件

```
[root@localhost ~]# rpm -ql iptables-services
```

- 保存规则

```
[root@localhost ~]# iptables-save > /etc/sysconfig/iptables
```

- 重载

```
[root@localhost ~]#iptables-restore < /etc/sysconfig/iptables
```

- 基本配置

```
iptables -F
#删除现有规则
```

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
#配置默认链策略
```

案例：白名单

```
[root@localhost ~]# iptables -t filter -F
[root@localhost ~]# iptables -P INPUT DROP
[root@localhost ~]# iptables -t filter -I INPUT -p tcp --dport=22 -j ACCEPT
[root@localhost ~]# iptables -t filter -I INPUT -p tcp --dport=80 -j ACCEPT
```

案例：黑名单

```
[root@localhost ~]# iptables -P INPUT ACCEPT
[root@localhost ~]# iptables -F
[root@localhost ~]# iptables -t filter -A INPUT -s 192.168.2.20/24 -p tcp --dport 80 -j
DROP
```

案例：通过lo访问本机数据

```
[root@localhost ~]# iptables -I INPUT -d 127.0.0.1 -p tcp --dport=9000 -i lo -j ACCEPT

[root@localhost ~]# iptables -I INPUT -i lo -j ACCEPT
#允许通过本地回环网卡访问本机
```

案例：允许连接态产生衍生态

```
[root@localhost ~]# iptables -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

iptables filter表应用案例

案例1：

```
[root@localhost ~]# yum -y install httpd vsftpd sshd
[root@localhost ~]# systemctl start httpd vsftpd sshd

[root@localhost ~]# iptables -t filter -F
[root@localhost ~]# iptables -I INPUT -p tcp --dport 80 -j ACCEPT
[root@localhost ~]# iptables -I INPUT -p tcp --dport 20:21 -j ACCEPT
[root@localhost ~]# iptables -I INPUT -p tcp --dport 22 -j ACCEPT
[root@localhost ~]# iptables -A INPUT -j REJECT

#在存问题
本机无法访问本机
例如：ping 127.0.0.1
解决方法
[root@localhost ~]# iptables -I INPUT -i lo  -j ACCEPT

本机无法访问其它主机
例如：ssh remote_host
解决方法
[root@localhost ~]# iptables -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

FTP无法访问
解决方法1：
[root@localhost ~]# iptables -I INPUT -p tcp --dport 20:21 -j ACCEPT
[root@localhost ~]# vim /etc/vsftpd/vsftpd.conf
pasv_min_port=50000
pasv_max_port=60000
[root@localhost ~]# iptables -I INPUT -p tcp --dport 50000:60000 -j ACCEPT

解决方法2：使用连接追踪模块
[root@localhost ~]# iptables -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
[root@localhost ~]# iptables -I INPUT -p tcp --dport 20:21 -j ACCEPT
[root@localhost ~]# modprobe nf_conntrack_ftp #临时方法,添加连接追踪模块

[root@localhost ~]# vim /etc/sysconfig/iptables-config
IPTABLES_MODULES="nf_conntrack_ftp"
#启动服务时加载
#针对数据端口连接时，将三次握手第一次状态由NEW识别为RELATED
```

案例2：iptables标准流程

```
[root@localhost ~]# iptables -F
[root@localhost ~]# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
[root@localhost ~]# iptables -A INPUT -i lo -j ACCEPT
[root@localhost ~]# iptables -A INPUT -s 192.168.2.0/24 -j ACCEPT #允许内网任何访问
[root@localhost ~]# iptables -A INPUT -p tcp --syn --dport 80 -j ACCEPT
[root@localhost ~]# iptables -A INPUT -p tcp --syn --dport 22 -j ACCEPT
[root@localhost ~]# iptables -A INPUT -p tcp --syn --dport 21 -j ACCEPT
[root@localhost ~]# iptables -A INPUT -j REJECT
[root@localhost ~]# modprobe nf_conntrack_ftp

```

```
[root@localhost ~]# iptables-save > /etc/sysconfig/iptables
[root@localhost ~]# vim /etc/sysconfig/iptables-config
IPTABLES_MODULES="nf_conntrack_ftp"
```

案例3：扩展匹配

```
#-m icmp
[root@localhost ~]# iptables -F
[root@localhost ~]# iptables -t filter -I INPUT -p icmp -m icmp --icmp-type echo-reply -j ACCEPT #允许ping回应
[root@localhost ~]# iptables -A INPUT -j REJECT


#-m iprange
[root@localhost ~]# iptables -t filter -I INPUT -m iprange --src-range 192.168.2.10-192.168.2.100 -j REJECT

#-m multiport
[root@localhost ~]# iptables -t filter -I INPUT -p tcp -m multiport --dports 20,21,22,25,80,110 -j ACCEPT

#-m tos 根据ip协议头部 type of service进行过滤
[root@localhost ~]# iptables -F
[root@localhost ~]# tcpdump -i eth0 -nn port 22 -vvv
 #抓取远程主机访问本机的ssh数据包，观察TOS值
[root@localhost ~]# tcpdump -i eth0 -nn port 22 -vvv
 #抓取远程从本机rsync或scp复制文件，观察TOS值
 #ssh: tos 0x0  0x10
 #scp: tos 0x0  0x8
 #rsync: tos 0x0 0x8
[root@localhost ~]# iptables -t filter -A INPUT -p tcp --dport 22 -m tos --tos 0x10 -j ACCEPT
[root@localhost ~]# iptables -t filter -A INPUT -j REJECT


#-m tcp 按TCP控制位进行匹配
Flags:SYN ACK FIN RST URG PSH  ALL NONE
```

```
[root@localhost ~]# iptables -t filter -A INPUT -p tcp -m tcp --tcp-flags SYN,ACK,FIN,RST SYN --dport 80 -j ACCEPT
[root@localhost ~]# iptables -t filter -A INPUT -p tcp --syn --dport 80 -j ACCEPT
#--tcp-flags SYN,ACK,FIN,RST SYN 检查四个标记位，只有SYN标记位才匹配，即只允许三次握手中的第一次握手，等价于--syn

#-m comment 对规则进行备注说明
[root@localhost ~]# iptables -A INPUT -s 192.168.2.250 -m comment --comment "cloud host" -j REJECT

#-m mark 使用mangle表的标记方法,配合mangle表使用
[root@localhost ~]# iptables -t filter -A INPUT -m mark 2 -j REJECT

```

案例4：扩展动作

```
#-j LOG 记录至日志中
[root@localhost ~]# grep 'kern.*' /etc/rsyslog.conf
kern.* /var/log/kernel.log
[root@localhost ~]# systemctl restart rsyslog
[root@localhost ~]# iptables -j LOG -h
[root@localhost ~]# iptables -t filter -A INPUT -p tcp --syn --dport 22 -j LOG --log-prefix " localhost_ssh "
[root@localhost ~]# iptables -t filter -A INPUT -p tcp --syn --dport 22 -j ACCEPT
[root@localhost ~]# iptables -t filter -A INPUT -j REJECT

#-j REJECT
当访问一个未开启的TCP端口时，应该返回一个带有RST标记的数据包
当访问一个未开启的UDP端口，结果返回port xxx unreachable
当访问一个开启的TCP端口，但被防火墙REJECT，结果返回port xxx unreachable
[root@localhost ~]# iptables -j REJECT -h
[root@localhost ~]# iptables -t filter -A INPUT -p tcp --dport 22 -j REJECT --reject-with tcp-reset //返回一个自定义消息类型


#-j MARK 进行标记，可在LVS调度器中应用
[root@localhost ~]# iptables -t mangle -L
[root@localhost ~]# iptables -j MARK -h
[root@localhost ~]# iptables -t mangle -A PREROUTING -s 192.168.2.110 -j MARK --set-mark 1
[root@localhost ~]# iptables -t mangle -A PREROUTING -s 192.168.2.25 -j MARK --set-mark 2
[root@localhost ~]# iptables -t filter -A INPUT -m mark --mark 1 -j ACCEPT //按照标记匹配
[root@localhost ~]# iptables -t filter -A INPUT -m mark --mark 2 -j REJECT
```

iptables nat表应用案例

- nat表作用

  导流

- nat表作用位置

  KVM或OpenStack中虚拟机或云主机与外部通信

  Docker管理的容器与外部通信

- nat表规则动作所对应的链

  SNAT 源地址转换 应用于出口POSTROUTING

  MASQUERADE 源地址转换 应用于出口POSTROUTING

  DNAT 目标地址转换 应用于进口PREROUTING

  REDIRECT 端口重定向 应用于进口PREROUTING

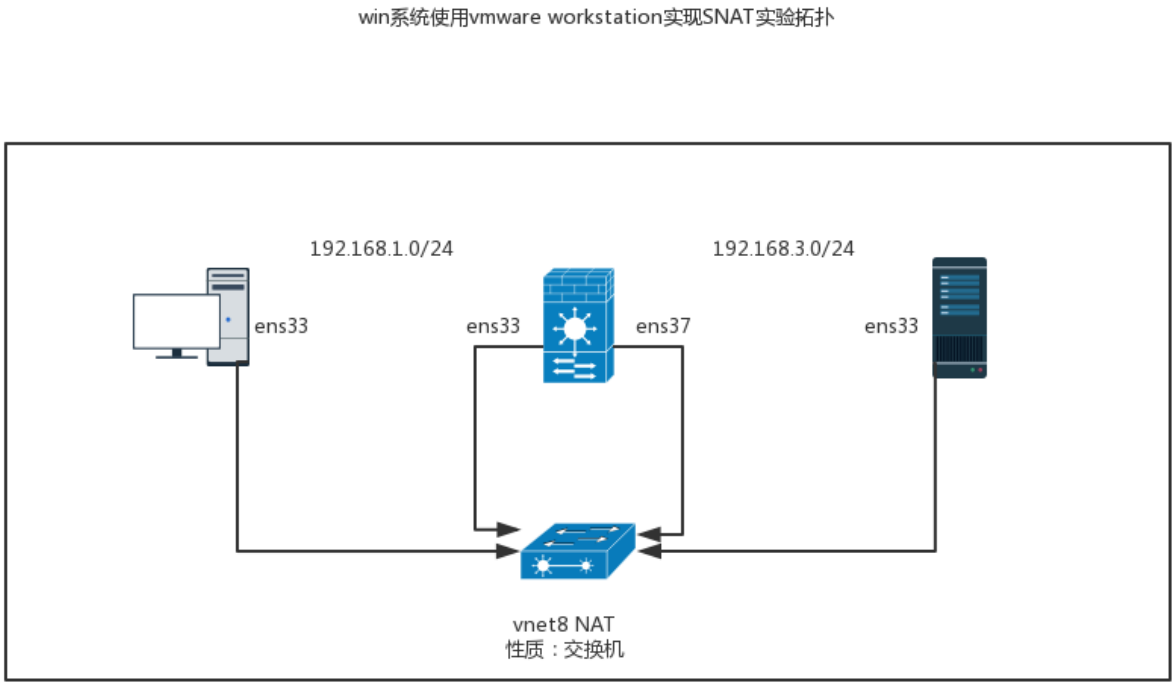
- 开启路由转发功能

```
[root@localhost ~]# sysctl -a | grep ip_forward
net.ipv4.ip_forward = 1
[root@localhost ~]# cat /proc/sys/net/ipv4/ip_forward
1
#以上为开启

#以下为关闭
[root@localhost ~]# echo 0 > /proc/sys/net/ipv4/ip_forward
[root@localhost ~]# cat /proc/sys/net/ipv4/ip_forward
0
```

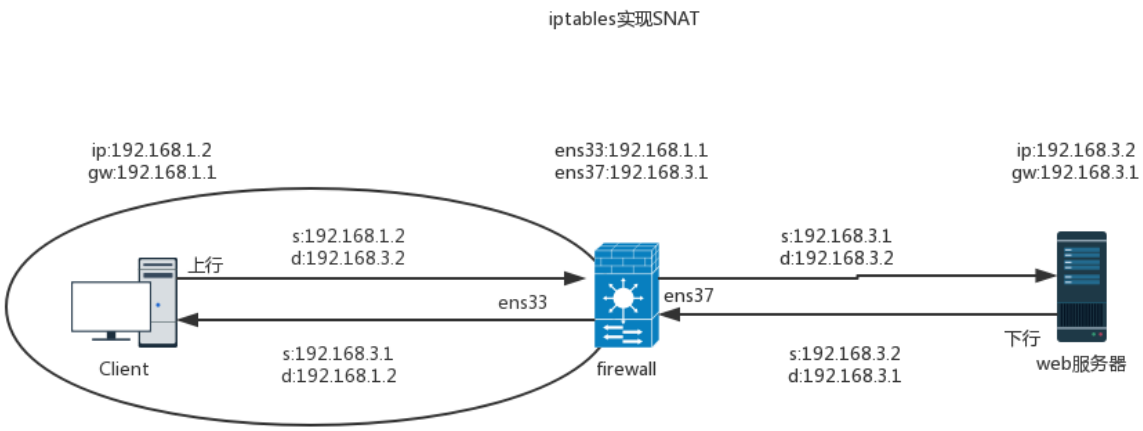
- SNAT,MASQUERAED 源地址转换案例

案例1：实现内网主机上网功能

- 网络连接拓扑



- 实现拓扑



- 配置命令
  - client主机配置

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens33
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
UUID=486d5c6d-17ed-4a3f-baab-92d56d042796
DEVICE=ens33
ONBOOT=yes
IPADDR=192.168.1.2
PREFIX=24
GATEWAY=192.168.1.1 #网关是firewall主机ens33接口ip
```

- firewall主机配置

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens33
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
UUID=4ac0aefc-628f-4461-ab59-636aae59965f
DEVICE=ens33
ONBOOT=yes
IPADDR=192.168.1.1
PREFIX=24

[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens37
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
```

```
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens37
DEVICE=ens37
ONBOOT=yes
IPADDR=192.168.3.1
PREFIX=24

```

```
#开启路由转发功能
[root@localhost ~]# cat /etc/sysctl.conf
# sysctl settings are defined through files in
# /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/.
#
# Vendors settings live in /usr/lib/sysctl.d/.
# To override a whole file, create a new file with the same in
# /etc/sysctl.d/ and put new settings there. To override
# only specific settings, add a file with a lexically later
# name in /etc/sysctl.d/ and put new settings there.
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
net.ipv4.ip_forward = 1

[root@localhost ~]# sysctl -p
net.ipv4.ip_forward = 1
```

```
[root@localhost ~]# iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ens37 -j SNAT --to-source 192.168.3.1
或
[root@localhost ~]# iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ens37 -j MASQUERADE
```

```
[root@localhost ~]# cat /etc/sysconfig/iptables
# sample configuration for iptables service
# you can edit this manually or use system-config-firewall
# please do not ask us to add additional ports/services to this default configuration
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
```

```
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
#-A FORWARD -j REJECT --reject-with icmp-host-prohibited
# 要注释
COMMIT


[root@localhost ~]# iptables-restore < /etc/sysconfig/iptables
[root@localhost ~]# iptables -t filter -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0            state
RELATED,ESTABLISHED
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            state NEW tcp dpt:22
REJECT     all  --  0.0.0.0/0            0.0.0.0/0            reject-with icmp-
host-prohibited

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

```
[root@localhost ~]# iptables -t nat -nL
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
RETURN     all  --  192.168.122.0/24     224.0.0.0/24
RETURN     all  --  192.168.122.0/24     255.255.255.255
MASQUERADE  tcp  --  192.168.122.0/24    !192.168.122.0/24     masq ports: 1024-
65535
MASQUERADE  udp  --  192.168.122.0/24    !192.168.122.0/24     masq ports: 1024-
65535
MASQUERADE  all  --  192.168.122.0/24    !192.168.122.0/24
MASQUERADE  all  --  192.168.1.0/24       0.0.0.0/0     #配置后的效果
```

- web服务器配置

  在配置其它IP地址前，请先安装httpd服务，便于验证结果

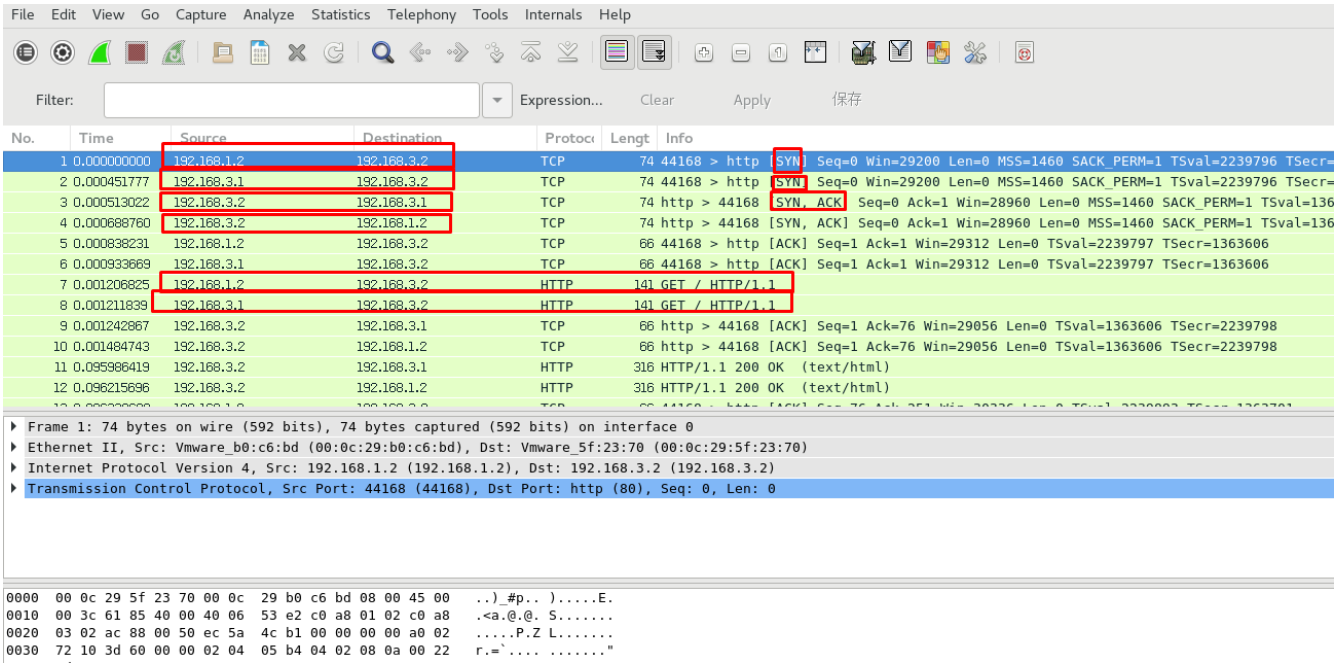
```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens33
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
UUID=bb300759-8c34-4e8a-b708-089b105425c3
DEVICE=ens33
ONBOOT=yes
IPADDR=192.168.3.2
PREFIX=24
GATEWAY=192.168.3.1 #网关为firewall主机ens37接口IP地址
```

- client主机结果验证

```
[root@localhost ~]# curl http://192.168.3.2
```

- 在firewall主机或web服务器安装wireshark抓包验证

```
[root@localhost ~]# yum -y install wireshark*
```



案例2：实现KVM虚拟机访问外部主机

路线：kvm_instance(192.168.122.0/24)--->192.168.122.1 virbr0(kvm虚拟机网关）ens33 192.168.2.10--->192.168.2.20(外部主机）

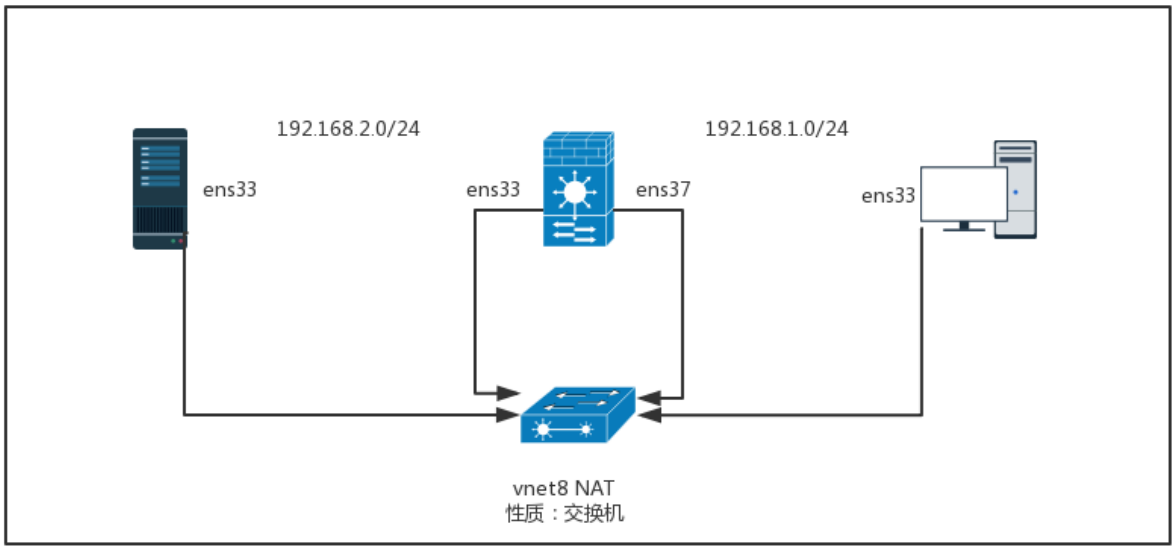
```
#方法一
[root@localhost ~]# iptables -t nat -F

[root@localhost ~]# iptables -t nat -A POSTROUTING -s 192.168.122.0/24 -j SNAT --to 192.168.2.10

[root@localhost ~]# iptables -t nat -A POSTROUTING -s 192.168.122.0/24 ! -d 192.168.122.0/24 -j SNAT --to 192.168.2.10

#方法二
[root@localhost ~]# iptables -t nat -F
[root@localhost ~]# iptables -t nat -A POSTROUTING -s 192.168.122.0/24 -j MASQUERADE
[root@localhost ~]# iptables -t nat -A POSTROUTING -s 192.168.122.0/24 ! -d 192.168.122.0/24 -j MASQUERADE
```

- DNAT 目标地址转换(端口映射)案例

案例1：实现局域网内发布服务器

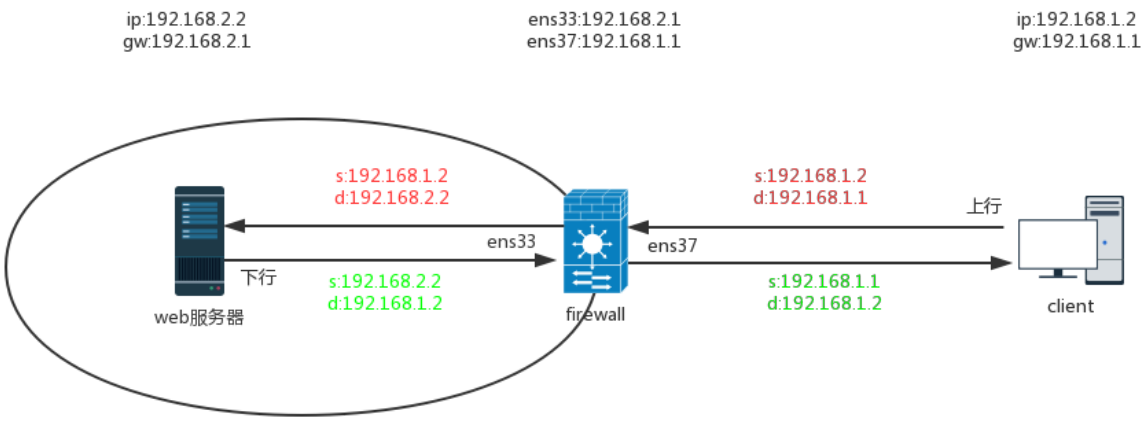
- 网络连接拓扑

win系统使用vmware workstation实现DNAT实验拓扑



- 实验拓扑

iptables实现DNAT



- 配置命令
  - web服务器配置

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens33
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
UUID=486d5c6d-17ed-4a3f-baab-92d56d042796
DEVICE=ens33
ONBOOT=yes
IPADDR=192.168.1.2
PREFIX=24
GATEWAY=192.168.1.1


[root@localhost ~]# yum -y install httpd wireshark*
[root@localhost ~]# cat /var/www/html/index.html
dnat test
[root@localhost ~]# systemctl start httpd
```

  - firewall主机配置

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens33
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
UUID=4ac0aefc-628f-4461-ab59-636aae59965f
DEVICE=ens33
ONBOOT=yes
```

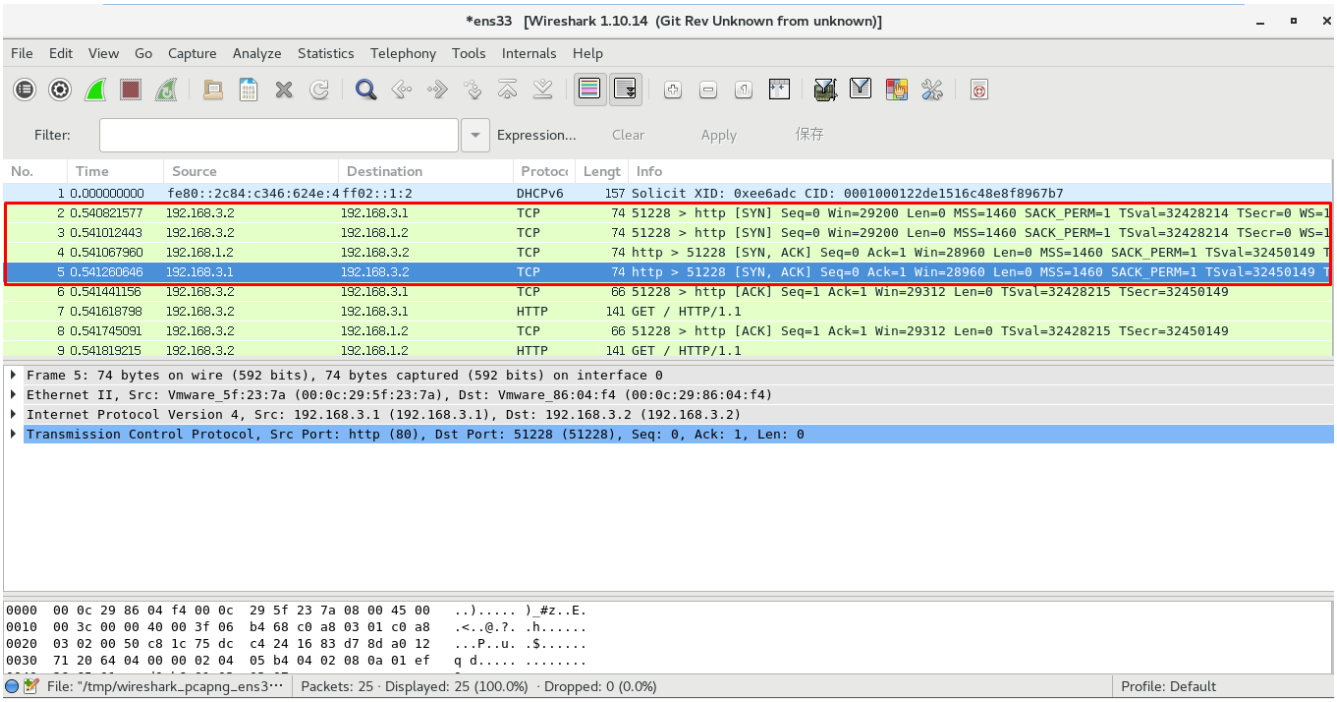
```
IPADDR=192.168.1.1
PREFIX=24

[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens37
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens37
DEVICE=ens37
ONBOOT=yes
IPADDR=192.168.3.1
PREFIX=24

[root@localhost ~]# cat /etc/sysctl.conf
net.ipv4.ip_forward = 1


[root@localhost ~]# iptables -t nat -A PREROUTING -d 192.168.3.1 -p tcp --dport 80
-j DNAT --to-destination 192.168.1.2


[root@localhost ~]# iptables -t nat -nL
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       tcp  --  0.0.0.0/0            192.168.3.1          tcp dpt:80
to:192.168.1.2

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
RETURN     all  --  192.168.122.0/24     224.0.0.0/24
RETURN     all  --  192.168.122.0/24     255.255.255.255
MASQUERADE  tcp  --  192.168.122.0/24    !192.168.122.0/24     masq ports: 1024-
65535
MASQUERADE  udp  --  192.168.122.0/24    !192.168.122.0/24     masq ports: 1024-
65535
MASQUERADE  all  --  192.168.122.0/24    !192.168.122.0/24
MASQUERADE  all  --  192.168.1.0/24       0.0.0.0/0
```

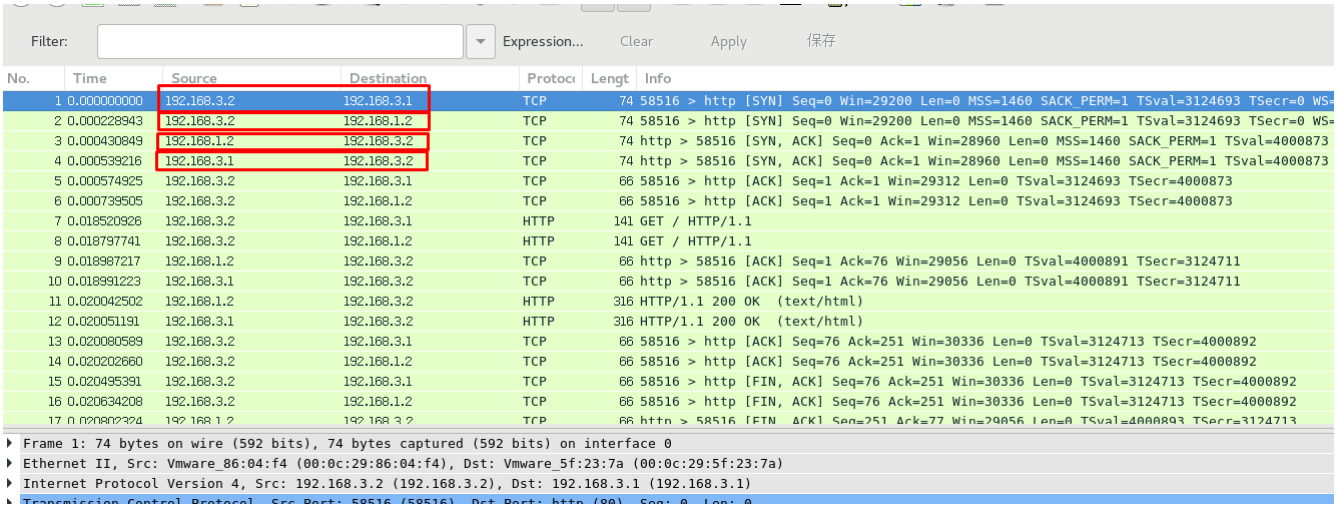
- client主机配置

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens33
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
UUID=bb300759-8c34-4e8a-b708-089b105425c3
DEVICE=ens33
ONBOOT=yes
IPADDR=192.168.3.2
PREFIX=24
GATEWAY=192.168.3.1
```

- client主机结果验证

```
[root@localhost ~]# curl http://192.168.3.1
dnat test
```





案例2：让外部主机可以访问KVM虚拟机（内网主机） PREROUTING 路由之前

建议：内网主机能访问外部主机（例如基于SNAT方式）

路线：kvm_instance(192.168.122.0/24) <--- 192.168.122.1 virbr0 ens33 192.168.2.10 <--- 192.168.2.20 外部主机

```
[root@localhost ~]# iptables -F
[root@localhost ~]# iptables -t nat -A PREROUTING -d 192.168.2.10 -p tcp --dport 80 -j DNAT --to 192.168.122.43:80
[root@localhost ~]# iptables -t nat -A PREROUTING -d 192.168.2.10 -p tcp --dport 2222 -j DNAT --to 192.168.122.43:22
[root@localhost ~]# iptables -t nat -A PREROUTING -d 192.168.2.10 -p tcp --dport 8080 -j DNAT --to 192.168.122.43:8080
```

思考：如果有两台内网服务器需要提供80/tcp服务，如何映射？

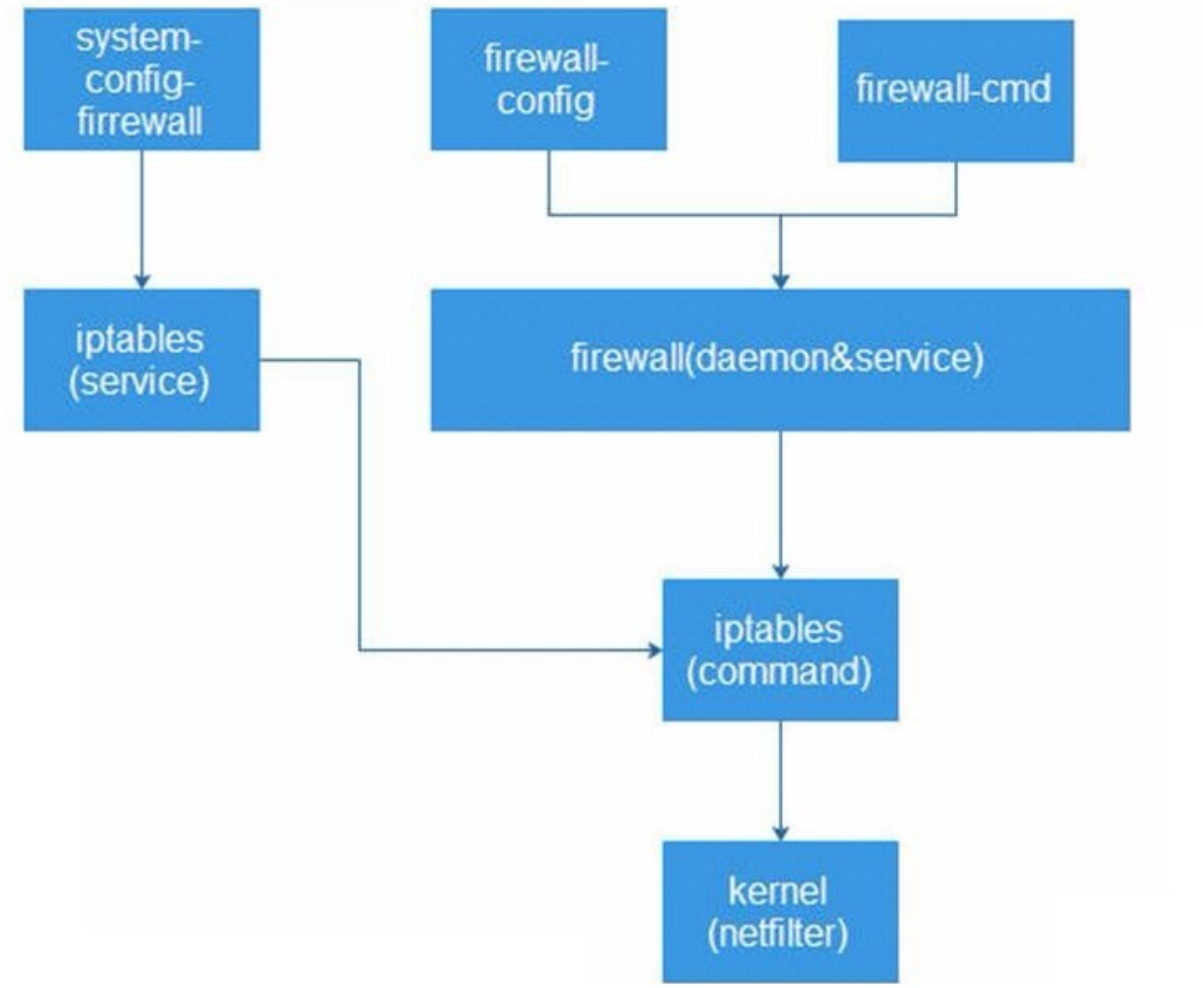
```
[root@localhost ~]# iptables -t nat -A PREROUTING -d 192.168.2.10 -p tcp --dport 80 -j DNAT --to 192.168.122.43:80
[root@localhost ~]# iptables -t nat -A PREROUTING -d 192.168.2.11 -p tcp --dport 80 -j DNAT --to 192.168.122.53:80
```

firewalld

Firewalld是什么？

FireWalld属于动态防火墙，是CentOS7系统中用于对netfilter内核模块用户空间管理工具。

FireWalld仅仅替代了iptables service部分，其底层还是使用iptables做为防火墙规则管理入口。



总结：

- 动态防火墙
- 用于管理netfilter用户空间的工具
- 调用了iptables命令

FireWalld中zone概念及作用

区域(zone)是firewalld预先准备好的防火墙策略集合，即可策略模板，用于可以根据不同的应用场景进行切换。

例如： 你有一台笔记本电脑，每天都要在公司办公室、咖啡厅和家使用。 我们来对场所进行安全性由高到低排序：家、公司办公室、咖啡厅。

我们希望为这台笔记本电脑指定如下防火墙策略规则： 在家中允许访问所有服务； 在公司办公室内仅允许访问文件共享服务； 在咖啡厅仅允许上网浏览。 在以往，我们需要频繁地手动设置防火墙策略规则，而现在只需要预设好区域集合， 随时都可以自动切换了，从而极大地提升了防火墙策略的应用效率。

FireWalld中zone分类

FireWalld不同区域之间的差异主要是每个区域对待数据包的默认行为不同

Firewalld默认共9个zone，分别为：

- block（拒绝）
- dmz（非军事化）
- drop（丢弃）
- external（外部）
- home（家庭）
- internal（内部）
- public（公开）Firewalld默认区域
- trusted（信任）
- work（工作区）

FireWalld文件

Firewalld文件分为两大类：

/usr/lib/firewalld/services:firewalld服务默认在此目录下定义了70多种服务，可以直接使用。

/usr/lib/firewalld/zones:区域配置文件

/etc/firewalld/zones:默认区域配置文件，配置文件中指定了编写完成的规则

firewalld文件作用：人性化管理规则；通过服务组织端口分组更加高效，如果一个服务使用若干网络端口，则服务的配置文件就相当于提供了到这些端口的规则管理的批量操作快捷方式。

Firewalld语法

命令语法：

```
firewall-cmd [--zone=zone] 动作 [--permanent]
```

> 如果不指定--zone选项，则为当前所在的默认区域，--permanent选项为是否将改动写入到区域配置文件中

Firewall的状态

```
[root@localhost ~]# firewall-cmd --state
running
#查看状态

[root@localhost ~]# firewall-cmd --reload
success
#重新加载防火墙，中断用户连接，临时配置清除掉，加载配置文件中的永久配置
```

```
[root@localhost ~]# firewall-cmd --complete-reload
success
#重新加载防火墙，不中断用户的连接（防火墙出严重故障时使用）

[root@localhost ~]# firewall-cmd --panic-on
#紧急模式，强制关闭所有网络连接
```

FireWalld中动作

动作中查看操作

```
[root@localhost ~]# firewall-cmd  xxx
--get-icmptypes      ##查看支持的所有ICMP类型
--get-zones          ##查看所有区域
--get-default-zone   ##查看当前的默认区域
--get-active-zones   ##查看当前正在使用的区域
--get-services       ##查看当前区域支持的服务
--list-services      ##查看当前区域开放的服务列表
--list-services --zone=home  ##查看指定域开放的服务列表
--list-all           ##查看默认区域内的所有配置，类似与iptables -L -n
--list-all-zones     ##查看所有区域所有配置
```

更改区域操作

```
[root@localhost ~]# firewall-cmd  xxx
--set-default-zone=work ##更改默认的区域
```

新建规则

新建 --add

```
[root@localhost ~]# firewall-cmd  xxx
--add-interface=eth0 ##将网络接口添加到默认的区域内
--add-port=12222/tcp --permanent ##添加端口到区域开放列表中
--add-port=5000-10000/tcp --permanent ##将端口范围添加到开放列表中；
--add-service=ftp --permanent ##添加服务到区域开放列表中（注意服务的名称需要与此区域支持的服务列表中的名称一致）
--add-source=192.168.1.1 ##添加源地址的流量到指定区域
--add-masquerade ##开启SNAT（源地址转换）
```

删除规则

删除 --remove

```
[root@localhost ~]# firewall-cmd  xxx
--remove-service=http ##在home区域内将http服务删除在开放列表中删除
--remove-interface=eth0 ##将网络接口在默认的区域内删除
--remove-source=192.168.1.1 ##删除源地址的流量到指定区域

```

改变规则

改变 change

```
[root@localhost ~]# firewall-cmd  xxx
--change-interface=eth1 ##改变指定的接口到其他区域
```

查询规则

查询 query

```
[root@localhost ~]# firewall-cmd  xxx
--query-masquerade ##查询SNAT的状态
--query-interface=eth0 ##确定该网卡接口是否存在于此区域
```

端口转发

端口转发可以将指定地址访问指定的端口时，将流量转发至指定地址的指定端口。转发的目的如果不指定ip的话就默认为本机，如果指定了ip却没指定端口，则默认使用来源端口。

注：以下部分可能需要2台主机完成。建议：最好先画图。

```
#通过firewalld实现SNAT
[root@localhost ~]# firewall-cmd --add-masquerade --permanent
[root@localhost ~]# firewall-cmd --reload


# 将80端口的流量转发至8080
#firewall-cmd --add-forward-port=port=80:proto=tcp:toport=8080



# 将80端口的流量转发至
firewall-cmd --add-forward-port=port=80:proto=tcp:toaddr=192.168.2.20

#删除
[root@localhost ~]# firewall-cmd --remove-forward-
port=port=80:proto=tcp:toaddr=192.168.2.20 --permanent

```

```
# 将80端口的流量转发至192.168.2.20的8080端口
firewall-cmd --add-forward-port=port=80:proto=tcp:toaddr=192.168.2.20:toport=8080

```

如果配置好端口转发之后不能用，可以检查下面两个问题：

- 比如将80端口转发至8080端口，首先检查本地的80端口和目标的8080端口是否开放监听了
- 其次检查是否允许伪装IP，没允许的话要开启伪装IP

Rich规则

当基本firewalld语法规则不能满足配置要求时，可以使用rich规则来完成更加复杂的功能。

Rich规则帮助

```
[root@localhost ~]# man 5 firewalld.richlanguage
```

Rich规则选项

```
--add-rich-rule='rule'          ##新建rich规则
--remove-rich-rule='rule'       ##删除rich规则
--query-rich-rule='rule'        ##查看单条rich规则
--list-rich-rules               ##查看rich规则列表
```

Rich规则案例

- 拒绝某一主机访问

```
[root@localhost ~]# firewall-cmd --permanent --zone=public --add-rich-rule='rule family=ipv4 source address=192.168.2.20/32 reject'
[root@localhost ~]# firewall-cmd --reload
```

- 抛弃icmp协议所有数据包

```
[root@localhost ~]# firewall-cmd --permanent --add-rich-rule='rule protocol value=icmp drop'
success

[root@localhost ~]# firewall-cmd --reload
success

#删除
[root@localhost ~]# firewall-cmd --permanent --remove-rich-rule='rule protocol value=icmp drop'
```

- 允许某一网段一段端口通过

```
[root@localhost ~]# firewall-cmd --permanent --zone=public --add-rich-rule='rule family=ipv4 source address=192.168.2.0/24 port port=7900-7905 protocol=tcp accept'
success
```

- 开启SNAT

```
[root@localhost ~]# firewall-cmd --permanent --add-rich-rule='rule family=ipv4 source address=192.168.1.0/24 masquerade'
```

- 端口转发

```
[root@localhost ~]# firewall-cmd --add-masquerade --permanent
[root@localhost ~]# firewall-cmd --permanent --add-rich-rule='rule family=ipv4 source address=192.168.2.10/24 forward-port port=80 protocol=tcp to-port=8090 to-addr=192.168.2.20'
[root@localhost ~]# firewall-cmd --reload
```