firewall

1. 防火墙的入门知识
从逻辑上讲防火墙可以分为主机防火墙和网络防护墙。
主机防火墙：针对个别主机对出站入站的数据包进行过滤。（操作对象为个体）
网络防火墙：处于网络边缘，针对网络入口进行防护。（操作对象为整体）
从物理上讲防火墙可以分为硬件防火墙和软件防火墙。
硬件防火墙：通过硬件层面实现防火墙的功能，性能高，成本高。
软件防火墙：通过应用软件实现防火墙的功能，性能低，成本低。

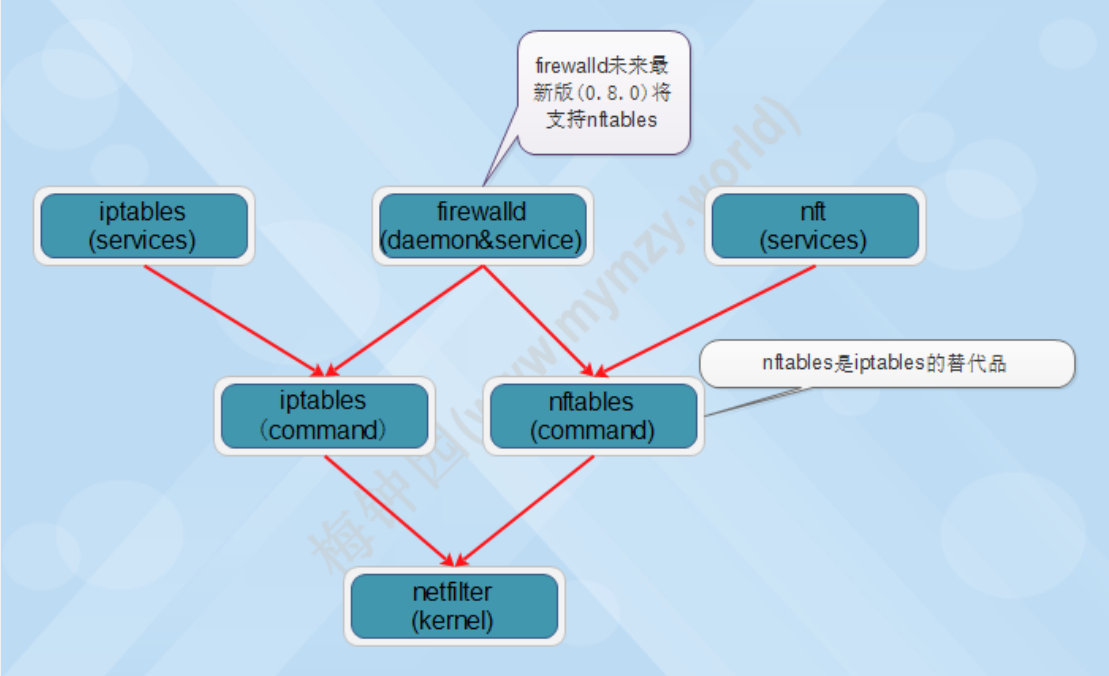
2. 系统防火墙发展过程
防火墙的发展史就是从墙到链再到表，也是从简单到复杂的过程。
防火墙工具变化如下：
ipfirewall--->ipchains--->iptables-->nftables
Linux 2.0 版内核中，包过滤机制为 ipfw，管理工具是 ipfwadm。
Linux 2.2 版内核中，包过滤机制为 ipchain，管理工具是 ipchains。
Linux 2.4，2.6,3.0+版内核中，包过滤机制为 netfilter，管理工具是 iptables。
Linux 3.1（3.13+）版内核中，包过滤机制为 netfilter，中间采取 daemon 动态管理防火墙，管理工具是 firewalld。目前低版本的 firewalld 通过调用 iptables(command)，它可以支持老的 iptables 规则（在 firewalld 里面叫做直接规则），同时 firewalld 兼顾了 iptables,ebtables,ip6tables 的功能。

iptables 和 nftables
　　nftables 诞生于 2008 年，2013 年底合并到 Linux 内核，从 Linux 3.13 起开始作为 iptables 的替代品提供给用户，它是新的数据包分类框架，新的 linux 防火墙管理程序，旨在替代现存的 {ip,ip6,arp,eb}_tables，它的用户空间管理工具是 nft。由于 iptables 的一些缺陷，目前正在慢慢过渡用 nftables 替换 iptables，同时由于这个新的框架的兼容性，所以 nftables 也支持在这个框架上运行直接 iptables 这个用户空间的管理工具。
　　nftables 实现了一组被称为表达式的指令，可通过在寄存器中储存和加载来交换数据。也就是说，nftables 的核心可视为一个虚拟机，nftables 的前端工具 nft 可以利用内核提供的表达式去模拟旧的 iptables 匹配，维持兼容性的同时获得更大的灵活性。
　　而未来最新的 firewalld（0.8.0）默认使用将使用 nftables。详情可以看 www.firewalld.org

iptables、nftables 和 firewalld 之间的区别与联系

firewalld 同时支持 iptables 和 nftables，未来最新版本(0.8.0)默认将使用 nftables。简单的说 firewalld 是基于 nftfilter 防火墙的用户界面工具。而 iptables 和 nftables 是命令行工具。
firewalld 引入区域的概念，可以动态配置，让防火墙配置及使用变得简便。
准确说：iptables(command)的最底层是 netfilter，它的用户空间管理工具是 iptables
nftables(command)是 iptables(command) 的一个替代品并兼容 iptables(command)，最底

层依然是 netfilter，它的用户空间管理工具是 nft，同时未来 firewalld 最新版（0.8.0）也将默认支持 nftables(command)。https://firewalld.org/



iptables 会把配置好的防火墙策略交给内核层的 netfilter 网络过滤器来处理
firewalld 会把配置好的防火墙策略交给内核层的 nftables 包过滤框架来处理

3. centos6.X 到 centos7.X
centos6.X：防火墙由 netfilter 和 iptables 构成。其中 iptables 用于制定规则，又被称为防火墙的用户态；而 netfilter 实现防火墙的具体功能，又被称为内核态。简单地讲，iptables 制定规则，而 netfilter 执行规则。
centos7.X：防火墙在 6.X 防火墙的基础之上提出了新的防火墙管理工具，提出了区域的概念，通过区域定义网络链接以及安全等级。

4.怎样学好防火墙的配置
OSI7 层模型以及不同层对应哪些协议？
TCP/IP 三次握手，四次断开的过程，TCP HEADER，状态转换
常用的服务端口要非常清楚了解。
常用服务协议原理 http 协议，icmp 协议。

企业中安全配置原则：
尽可能不给服务器配置外网 IP，可以通过代理转发或者通过防火墙映射。
并发不是特别大情况有外网 IP，可以开启防火墙服务。
大并发的情况，不能开 iptables，影响性能，利用硬件防火墙提升架构安全。

iptables

四表五链和一些基础知识

表	链	作用
Filter	防火墙安全过滤功能	
	INPUT	指定到本地套接字的包，即到达本地防火墙服务器的数据包 **外面---->（门）房子 iptables**
	FORWARD	路由穿过的数据包，即经过本地防火墙服务器的数据包 **外面-----（前门）房子（后门）---房子**
	OUTPUT	本地创建的数据包 **外面<-----（门）房子 iptables**
NAT	将数据包中 IP 地址或者端口信息，内网到外网进行改写/外网到内网进行改写	
	PREROUTING	**一进来就对数据包进行改变，在路由之前**，进行数据包 IP 地址或端口信息的转换
	OUTPUT	本地创建的数据包在路由之前进行改变，本地防火墙要出去的流量进行相应转换（，用得不多，了解即可）
	POSTROUTING	**在数据包即将出去时改变数据包信息，在路由之后**，进行数据包 IP 地址或端口信息的转换
Mangle	对数据进行标记；拆解报文，做出修改，封装报文。	
	PREROUTING	基础使用上用得不多，可以不用管，了解即可
	INPUT	
	FORWARD	
	OUTPUT	
	POSTROUTING	
raw	关闭 nat 表上启用的连接追踪机制；忽略不计。 1)前端有承载大量并发请求连接时，一定不要开启连接追踪功能，不然内存就可能会溢出，前端大量连接请求遭到拒绝。 2)连接追踪需要记录，就需要内存空间，然后就可能溢出，一旦溢出，新连接就不能进来了。 3)如果不得不启用连接追踪，那么就必须有足够大的物理内存，并且给连接追踪能使用的最大内存调大。	
	PREROUTING	基础使用上用得不多，可以不用管，了解即可
	OUTPUT	

每个功能需要使用多个链，一个链上有 N 条规则，这样就相当于一个表了。这就是叫做 iptables 的原因。

流入：PREROUTING --> INPUT #外部到本机
流出：OUTPUT --> POSTROUTING #本机到外部

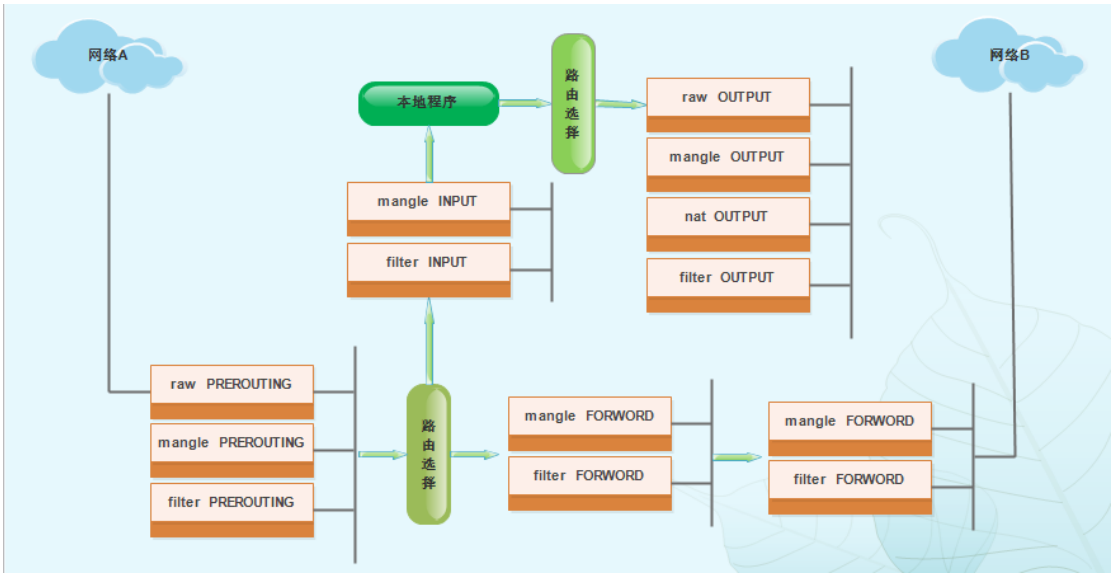
转发：PREROUTING --> FORWARD --> POSTROUTING #外部经过本机转发到其他主机

防火墙匹配规则流程

1. 防火墙是层层过滤的，实际是按照配置规则的顺序从上到下，从前到后进行过滤的。
2. 如果匹配上规则，即明确表示是阻止还是通过，数据包就不再向下匹配新的规则。
3. 如果规则中没有明确表明是阻止还是通过的，也就是没有匹配规则，
   向下进行匹配，直到匹配默认规则得到明确的阻止还是通过。
4. 防火墙的默认规则是所有规则执行完才执行的。

注意：iptables 只能针对 IPV4 地址，不能用来过滤 IPV6.

下图为 iptables 规则链匹配流程



iptables 命令基础使用

规则链管理

-F：flush，清空规则链；省略链，表示清空指定表上的所有的链；
-X：drop，删除用户自定义的空的规则链；
-Z：zero，清零，置零规则计数器；
-N：new，创建新的自定义规则链；
-P：Policy，为指定链设置默认策略；
对 filter 表中的链而言，默认策略通常有 ACCEPT（允许），DROP（丢弃），
REJECT（拒绝，挑衅行为式的拒绝）； deny drop reject
-E: rEname，重命令自定义链；当引用计数不为 0 的自定义链，无法改名，也无法删除；

规则管理

-A：append，将新规则追加于指定链的尾部；
-I：insert，将新规则插入至指定链的指定位置；
-D：delete，删除指定链上的指定规则；
有两种指定方式：
(1) 指定匹配条件；
(2) 指定规则编号；
-R：replace，替换指定链上的指定规则；

查看规则

-L：list，列出指定链上的所有规则；
-n: numberic，以数字格式显示地址和端口号；　#这样就不会反解
-v: verbose，显示详细信息；
-vv, -vvv #v 越多显示越详细
--line-numbers：显示规则编号；
-x: exactly，显示计数器计数结果的精确值；

-t, --table table 指定表：filter、nat、mangle 和 raw，默认为 filter

基本匹配

[!] -s, --src, --source IP|Netaddr，检查报文中源 IP 地址是否符合此处指定的地址范围；
[!] -d, --dst, --destination IP|Netaddr，检查报文中源 IP 地址是否符合此处指定的地址范围；
-p, --protocol {tcp|udp|icmp}：检查报文中的协议，即 ip 首部中的 protocols 所标识的协议；
-i, --in-interface IFACE：数据报文的流入接口；仅能用于 PREROUTING, INPUT 及 FORWARD 链上；
-o, --out-interface IFACE：数据报文的流出接口；仅能用于 FORWARD, OUTPUT 及 POSTROUTING 链上；

扩展匹配之隐式扩展

-m macth_name --spec_options
隐式扩展：对-p protocol 指明的协议进行的扩展，可省略-m 选项；

-p tcp
    --dport PORT[-PORT]：目标端口，可以是单个端口或连续多个端口；
    --sport PORT[-PORT]： 源端口，可以是单个或者多个连续端口
    --tcp-flags LIST1 LIST2：
        检查 LIST1 所指明的所有标志位，且这其中，
        LIST2 所表示出的所有标记位必须为 1，而余下的必须为 0；
        没有 LIST1 中指明的，不作检查；
        SYN, ACK, FIN, RST, PSH, URG
        --tcp-flags SYN,ACK,FIN,RST SYN
    --syn:检查新建 TCP 连接是否为第一次请求，等价于下面的写法：
        --tcp-flags SYN,ACK,FIN,RST SYN
-p udp
    --dport
    --sport

-p icmp
    --icmp-type
    可用数字表示其类型：
        0：echo-reply
        8: echo-request

target

-j TARGET：jump 至指定的 TARGET
常见的有：
    ACCEPT: 接受
    DROP: 丢弃
    REJECT: 拒绝
    RETURN: 返回调用链
    REDIRECT：端口重定向
    LOG: 记录日志
    MARK：做防火墙标记
    DNAT：目标地址转换
    SNAT：源地址转换
    MASQUERADE：地址伪装
    自定义链：由自定义链上的规则进行匹配检查

扩展匹配之显式扩展（必须用-m 指定）

1、multiport 扩展

以离散方式定义多端口匹配，默认的-p sport|dport 是不支持多个离散端口匹配的；最多指定 15 个端口；

[!] --source-ports,--sports port[,port|,port:port]...：指明多个源端口；
[!] --destination-ports,--dports port[,port|,port:port]...：指明多个离散的目标端口；
[!] --ports port[,port|,port:port]...
也可以不使用上面的参数，只要指定-m multiport，直接就能使用-p dport|sport 22,24,25
iptables -A INPUT -s 10.0.0.9 -m multiport -p tcp --dport 22,24,25 -j DROP

2、iprange 扩展
指明连续的（但一般是不能扩展为整个网络）ip 地址范围时使用；

[!] --src-range from[-to]：指明连续的源 IP 地址范围；
[!] --dst-range from[-to]：指明连续的目标 IP 地址范围；

iptables -I INPUT -d 172.16.100.9 -p tcp -m multiport --dports 22:23,80 -m iprange --src-range 172.16.100.1-172.16.100.120 -j ACCEPT
iptables -I OUTPUT -s 172.16.100.9 -p tcp -m multiport --sports 22:23,80 -m iprange --dst-range 172.16.100.1-172.16.100.120 -j ACCEPT

3、string 扩展
检查报文中出现的字符串；
--algo {bm|kmp} #bm|kmp 分别是两种不同的字符串比较算法
bm = Boyer-Moore
kmp = Knuth-Pratt-Morris
[!] --string pattern

iptables -I OUTPUT -m string --algo bm --string 'movie' -j REJECT

4、time 扩展
根据报文到达的时间与指定的时间范围进行匹配；

--datestart --datestart YYYY[-MM[-DD[Thh[:mm[:ss]]]]] #起始日期
--datestop --datestop YYYY[-MM[-DD[Thh[:mm[:ss]]]]#结束日期

--timestart
--timestop

--monthdays
--weekdays

具体可以看帮助

几种常见时间概述与关系

全球 24 个时区的划分

相较于两地时间表，可以显示世界各时区时间和地名的世界时区表（World Time），就显得精密与复杂多了，通常世界时区表的表盘上会标示着全球 24 个时区的城市名称，但究竟这 24 个时区是如何产生的？

过去世界各地原本各自订定当地时间，但随着交通和电讯的发达，各地交流日益频繁，不同的地方时间，造成许多困扰，于是在西元 1884 年的国际会议上制定了全球性的标准时，明定以英国伦敦格林威治这个地方为零度经线的起点（亦称为本初子午线），并以地球由西向东每 24 小时自转一周 360°，订定每隔经度 15°，时差 1 小时。而每 15°的经线则称为该时区的中央经线，将全球划分为 24 个时区，其中包含 23 个整时区及 180°经线左右两侧的 2 个半时区。

就全球的时间来看，东经的时间比西经要早，也就是如果格林威治时间是中午 12 时，则中央经线 15°E 的时区为下午 1 时，中央经线 30°E 时区的时间为下午 2 时；反之，中央经线 15°W 的时区时间为上午 11 时，中央经线 30°W 时区的时间为上午 10 时。以台湾为例，台湾位于东经 121°，换算后与格林威治就有 8 小时的时差。如果两人同时从格林威治的 0°各往东、西方前进，当他们在经线 180°时，就会相差 24 小时，所以经线 180°被定为国际换日线，由西向东通过此线时日期要减去一日，反之，若由东向西则要增加一日。

CET

欧洲中部时间（Central European Time，CET）是比世界标准时间（UTC）早一个小时的时区名称之一。它被大部分欧洲国家和部分北非国家采用。冬季时间为 UTC+1，夏季欧洲夏令时为 UTC+2。

UTC

协调世界时间，又称世界标准时间或世界协调时间，简称 UTC，从英文“Coordinated Universal Time”／法文“Temps Universel Cordonné”而来。台湾采用 CNS 7648 的《资料元及交换格式–资讯交换–日期及时间的表示法》（与 ISO 8601 类似）称之为世界统一时间。中国大陆采用 ISO 8601-1988 的国标《数据元和交换格式信息交换日期和时间表示法》（GB/T 7408）中称之为国际协调时间。

UTC 是经过平均太阳时(以格林威治时间 GMT 为准)、地轴运动修正后的新时标以及以「秒」为单位的国际原子时所综合精算而成的时间，计算过程相当严谨精密，因此若以「世界标准时间」的角度来说，UTC 比 GMT 来得更加精准。其误差值必须保持在 0.9 秒以内，若大于 0.9 秒则由位于巴黎的国际地球自转事务中央局发布闰秒，使 UTC 与地球自转周期一致。所以基本上 UTC 的本质强调的是比 GMT 更为精确的世界时间标准，不过对于现行表款来说，GMT 与 UTC 的功能与精确度是没有差别的。

GMT

格林尼治标准时间（旧译格林尼治平均时间或格林威治标准时间；英文：Greenwich Mean Time，GMT），十七世纪，格林威治皇家天文台为了海上霸权的扩张计画而进行天体观测。1675 年旧皇家观测所(Old Royal Observatory) 正式成立，位于英国伦敦郊区，到了 1884 年决定以通过格林威治的子午线作为划分地球东西两半球的经度零度。观测所门口墙上有一个标志 24 小时的时钟，显示当下的时间，对全球而言，这里所设定的时间是世界时

间参考点，全球都以格林威治的时间作为标准来设定时间，这就是我们耳熟能详的「格林威治标准时间」(Greenwich Mean Time，简称 G.M.T.)的由来，标示在手表上，则代表此表具有两地时间功能，也就是同时可以显示原居地和另一个国度的时间。但由于地球自转不均匀不规则，导致 GMT 不精确，现在已经不再作为世界标准时间使用。

CST

在国内一般指北京时间，China Standard Time，又名中国标准时间，是中国的标准时间。在时区划分上，属东八区，比协调世界时早 8 小时，记为 UTC+8，与中华民国国家标准时间（旧称“中原标准时间”）、香港时间和澳门时间和相同。与格林尼治时间相差 8 小时。值得注意的是 CST 却同时可以代表如下 4 个不同的时区：
Central Standard Time (USA) UT-6:00
Central Standard Time (Australia) UT+9:30
China Standard Time UT+8:00
Cuba Standard Time UT-4:00
可见，CST 可以同时表示美国，澳大利亚，中国，古巴四个国家的标准时间。

DST

所谓「夏日节约时间」Daylight Saving Time（简称 D.S.T.），是指在夏天太阳升起的比较早时，将时钟拨快一小时，以提早日光的使用，在英国则称为夏令时间(Summer Time)。这个构想于 1784 年由美国班杰明·富兰克林提出来，1915 年德国成为第一个正式实施夏令日光节约时间的国家，以削减灯光照明和耗电开支。自此以后，全球以欧洲和北美为主的约 70 个国家都引用这个做法。目前被划分成两个时区的印度也正在商讨是否全国该统一实行夏令日光节约时间。欧洲手机上也有很多 GSM 系统的基地台，除了会传送当地时间外也包括夏令日光节约时间，做为手机的时间标准，使用者可以自行决定要开启或关闭。值得注意的是，某些国家有实施「夏日节约时间」的制度，出国时别忘了跟随当地习惯在表上调整一下，这可是机械表没有的功能设计哦！

关系

CET=UTC/GMT + 1 小时
CST=UTC/GMT +8 小时
CST=CET+9

例如： Thursday 03/17/2011 between 0:30am and 6:00am CET (UTC/GMT +1 hour)
CET = 03/17/2011 0:30am and 6:00am
UTC/GMT = 03/17/2011 1:30am and 7:00am
CST = 03/17/2011 9:30am and 15:00am

5、connlimit 扩展
根据每客户端 IP（也可以是地址块）做并发连接数数量匹配； #单 IP 并发访问限制，
--connlimit-above n：连接的数量大于 n

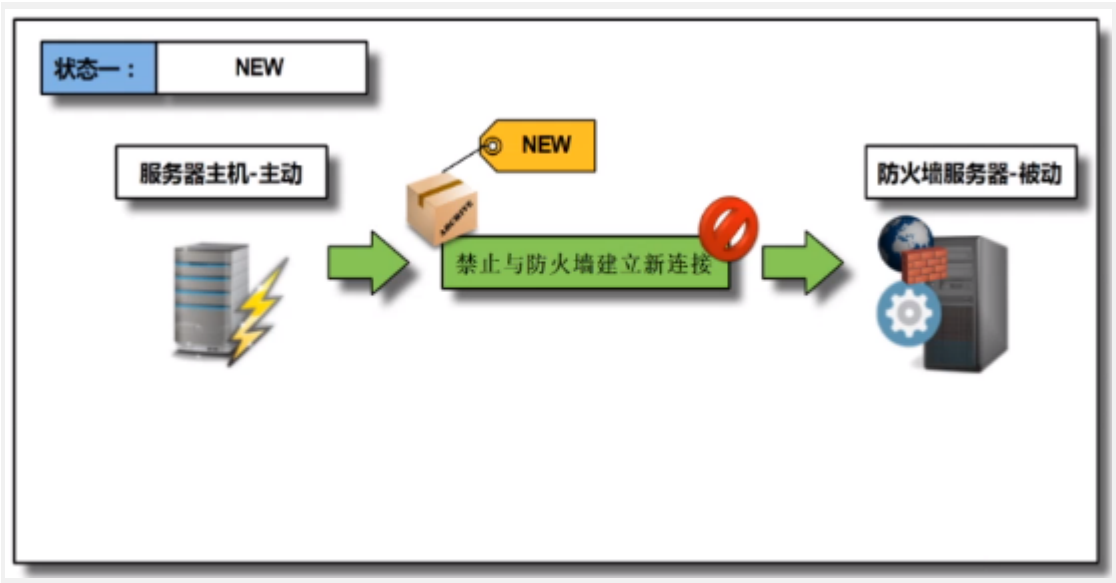
--connlimit-upto n: 连接的数量小于等于 n

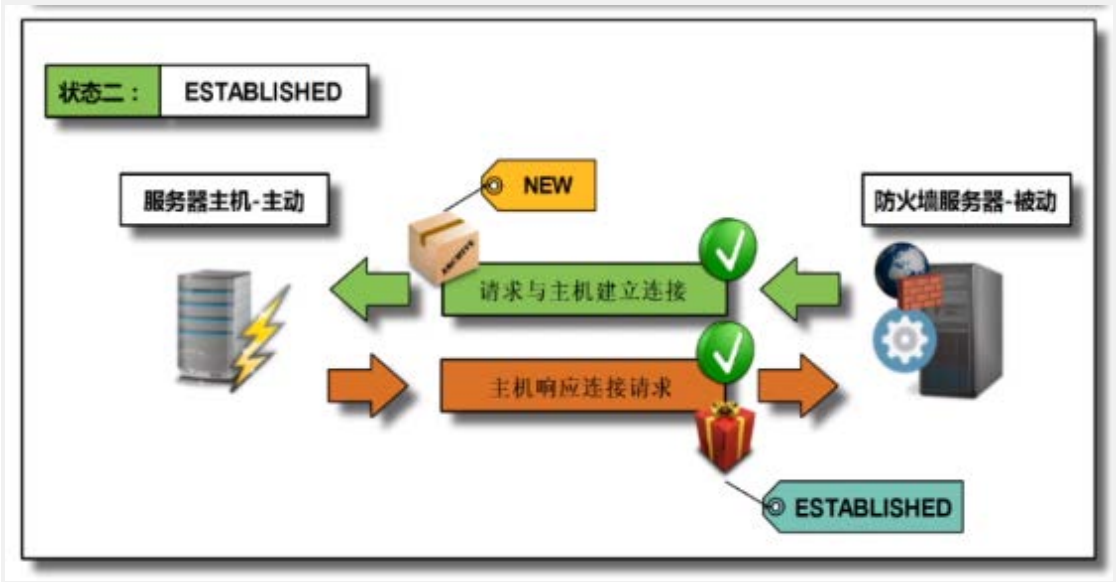
6、limit 扩展
基于收发报文的速率做检查；
令牌桶过滤器　　#使用的是令牌桶算法
--limit rate[/second|/minute|/hour|/day]　速率　　#limit 30/second
--limit-burst number　峰值　　#limit-burst 5

7、state 扩展
根据连接追踪机制检查连接的状态；
能够增强服务器（比如 web 服务）的安全性，避免反弹式木马攻击，但是必须打开连接追踪功能，这样会消耗服务器的性能。
反弹式木马其实就是被动等待肉鸡来连接的一种木马，如灰鸽子。一般防火墙对内部访问外部的数据都不会过问，肉鸡会主动连接控制端。

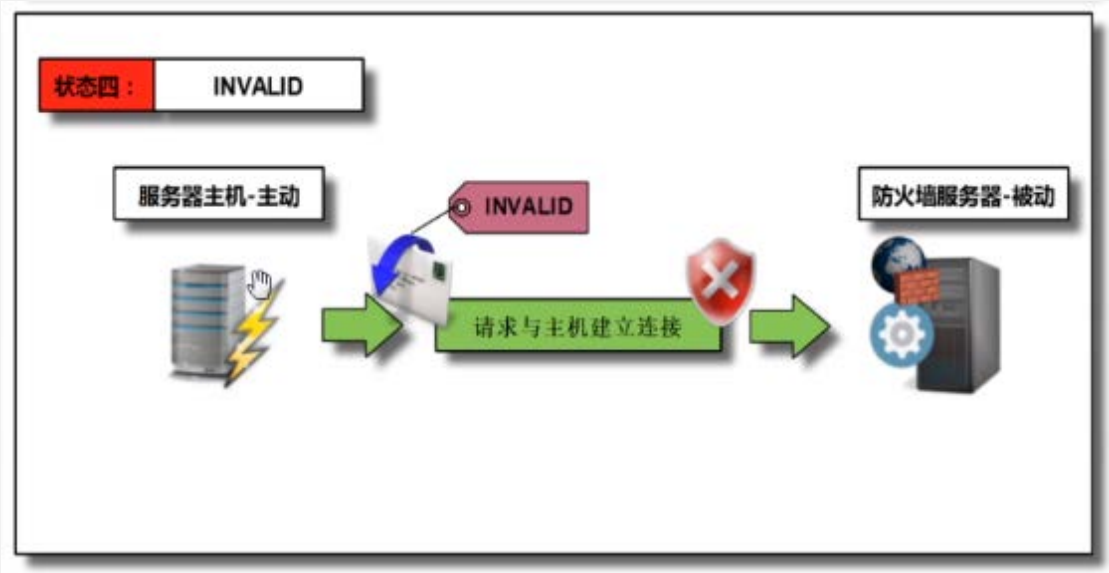
调整连接追踪功能所能够容纳的最大连接数量：
/proc/sys/net/nf_conntrack_max
已经追踪到并记录下的连接：
/proc/net/nf_conntrack
不同协议或连接类型追的时长：
/proc/sys/net/netfilter/
可追踪的连接状态：
NEW：新发出的请求；连接追踪模板中不存此连接相关的信息条目，因此，将其识别为第一次发出的请求；
ESTABLISHED：NEW 状态之后，连接追踪模板中为其建立的条目失效之前期间内所进行的通信的状态；
RELATED：相关的连接；如 ftp 协议的命令连接与数据连接之间的关系；
INVALIED：无法识别的连接；

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


状态一：
NEW
服务器主机-主动
NEW
防火墙服务器-被动
禁止与防火墙建立新连接


状态二：
ESTABLISHED
服务器主机-主动
NEW
防火墙服务器-被动
请求与主机建立连接
主机响应连接请求
ESTABLISHED


状态三：
RELATED
NEW
服务器主机-主动
防火墙服务器-被动
请求与主机建立连接
进行新的数据包通讯
RELATED
主机响应连接请求
ESTABLISHED



如何开放被动模式的 ftp 服务？

FTP 协议有两种工作方式：PORT 方式和 PASV 方式，中文意思为主动式和被动式。

Port 模式 :ftp server:tcp 21 <------client:dynamic　ftp server:tcp 20 ------>client:dynamic

Pasv 模式：ftp server:tcp 21 <----client:dynamic　ftp server:tcp dynamic <----client:dynamic

PORT（主动）方式的连接过程是 :客户端向服务器的 FTP 端口（默认是 21）发送连接请 求，服务器接受连接，建立一条命令链路。当需要传送数据时，客户 端在命令链路上用 PORT 命令告诉服务器："我打开了 XXXX 端口，你过来连接我"。于是服务器从 20 端口向客户端的 XXXX 端口发送连接请求，建立一条 数据链路来传送数据。

PASV（被动）方式的连接过程是 :客户端向服务器的 FTP 端口（默认是 21）发送连接请 求，服务器接受连接，建立一条命令链路。当需要传送数据时，服务 器在命令链路上用 PASV 命令告诉客户端："我打开了 XXXX 端口，你过来连接我"。于是客户端向服务器的 XXXX 端口发送连接请求，建立一条数据链路来 传送数据。

开放主动模式

```
iptables -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -s x.x.x.x --dport 20:21　-j ACCEPT
```

开放被动模式方法一：

如果用 vsftpd，修改配置文件，设置最小和最大打开端口

```
pasv_min_port=2222
pasv_max_port=2225
iptables -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -s x.x.x.x --dport 21　-j ACCEPT
iptables -A INPUT -p tcp -s x.x.x.x --dport 2222:2225　-j ACCEPT
```

方法二使用连接追踪模块：

(1) 装载 ftp 追踪时的专用的模块:

/lib/modules/2.6.32-696.el6.x86_64/kernel/net/netfilter/nf_conntrack_ftp

modinfo nf_conntrack_ftp.ko -- 查看模块信息
modprobe nf_conntrack_ftp -- 使用模块
(2) 放行请求报文：
命令连接：NEW, ESTABLISHED
数据连接：RELATED, ESTABLISHED
iptables -A INPUT -d LocalIP -p tcp --dport 21 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -d LocalIP -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
(3) 放行响应报文：
ESTABLISEHD
iptables -A OUTPUT -s LocalIP -p tcp -m state --state ESTABLISHED -j ACCEPT

8、如何保存及重载规则：
直接保存：规则自动保存到了/etc/sysconfig/iptables，用此命令保存的规则开机会自动生效。
iptables save
保存规则至指定文件：
iptables-save > /etc/sysconfig/iptables
从指定文件重载规则：
iptables-restore < /etc/sysconfig/iptables
保存规则：#iptables-save >/etc/iptables-script
恢复规则：#iptables-restore>/etc/iptables-script
开机自动恢复规则，把恢复命令添加到启动脚本：echo '/sbin/iptables-restore /etc/iptables-script' >>/etc/rc.d/rc.local

9、实现网络防火墙
实现网络防火墙最重要的就是要打开核心转发，并且按需求配置 FORWARD 链上的策略。
/etc/sysctl.conf #永久修改
查看核心转发功能是否开启：
cat /proc/sys/net/ipv4/ip_forward
sysctl net.ipv4.ip_forward

sysctl
-a 显示所有的系统参数
-w 临时改变某个指定参数的值
-p 从指定的文件加载系统参数，如不指定即从/etc/sysctl.conf 中加载

NAT

nat：Network Address Translation，安全性，网络层+传输层 #最初出现的目的是为了隐藏本地网络中的主机
proxy：代理，应用层。 #真正的请求者是代理服务器。

nat:

SNAT：只修改请求报文的源地址；

DNAT：只修改请求报文的目标地址；　#相对于请求报文来说

MASQUERADE：地址伪装，当外网 IP 不固定时

源地址转换：

iptables -t nat -A POSTROUTING -s LocalNET ! -d LocalNet -j SNAT --to-source ExtIP

iptables -t nat -A POSTROUTING -s LocalNET ! -d LocalNet -j MASQUERADE　#外网地址不是固定的

报文到达主机在 PREROUTING 后是不知道去哪里的，要等到确定了是转换（NAT）并确实了从哪个网卡发送出去（这里是指路由发生），再（路由发生后）在 POSTROUTING 的时候再做源地址转换。

目标地址转换：

iptables -t nat -A PREROUTING -d ExtIP -p tcp|udp --dport PORT -j DNAT --to-destination InterSeverIP[:PORT]

路由发生前，一律查 NAT 表，根据规则转换为能够在内网通信的地址。 #不然黄花菜都凉了

连接追踪：

iptables 的连接跟踪表最大容量为/proc/sys/net/ipv4/ip_conntrack_max，连接碰到各种状态的超时后就会从表中删除。

所以解决方法一般有两个：

(1) 加大 ip_conntrack_max 值

vi /etc/sysctl.conf

net.ipv4.ip_conntrack_max = 393216

net.ipv4.netfilter.ip_conntrack_max = 393216

(2): 降低 ip_conntrack timeout 时间

vi /etc/sysctl.conf

net.ipv4.netfilter.ip_conntrack_tcp_timeout_established = 300

net.ipv4.netfilter.ip_conntrack_tcp_timeout_time_wait = 120

net.ipv4.netfilter.ip_conntrack_tcp_timeout_close_wait = 60

net.ipv4.netfilter.ip_conntrack_tcp_timeout_fin_wait = 120

利用 iptables 的 recent 模块来抵御 DOS 攻击: 22，建立一个列表，保存有所有访问过指定的服务的客户端 IP

ssh: 远程连接，

iptables -I INPUT -p tcp --dport 22 -m connlimit --connlimit-above 3 -j DROP

iptables -I INPUT　-p tcp --dport 22 -m state --state NEW -m recent --set --name SSH

iptables -I INPUT -p tcp --dport 22 -m state --state NEW -m recent --update --seconds 300 --hitcount 3 --name SSH -j LOG --log-prefix "SSH Attach: "
iptables -I INPUT -p tcp --dport 22 -m state --state NEW -m recent --update --seconds 300 --hitcount 3 --name SSH -j DROP

1.利用 connlimit 模块将单 IP 的并发设置为 3；会误杀使用 NAT 上网的用户，可以根据实际情况增大该值；

2.利用 recent 和 state 模块限制单 IP 在 300s 内只能与本机建立 2 个新连接。被限制五分钟后即可恢复访问。
下面对最后两句做一个说明：
1.第二句是记录访问 tcp 22 端口的新连接，记录名称为 SSH
--set 记录数据包的来源 IP，如果 IP 已经存在将更新已经存在的条目

2.第三句是指 SSH 记录中的 IP，300s 内发起超过 3 次连接则拒绝此 IP 的连接。
--update 是指每次建立连接都更新列表；
--seconds 必须与--rcheck 或者--update 同时使用
--hitcount 必须与--rcheck 或者--update 同时使用

3.iptables 的记录：/proc/net/xt_recent/SSH
也可以使用下面的这句记录日志：
iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent --update --name SSH --second 300 --hitcount 3 -j LOG --log-prefix "SSH Attack"

iptables 七层访问过滤（知道有这个功能即可，不必深究）：

要实现需要做以下处理：
1）对内核中的 netfilter，打补丁 layer7，重新编译内核
2）对 iptables 打补丁，补上 layer7 模块，重启 iptables

为什么要使用 iptables 的七层过滤
作为网络管理员,对 P2P,QQ,酷狗,等软件是又爱又恨,大多数公司,为了提高工作效率禁止公司员工上 QQ,看视频等， 在市场上买专门的上网行为管理设备,随便都是好几 W,而使用 linux 来做网关,一样可以禁止 qq,酷狗等软件,成本才几千块,下面将介绍下怎么实现的
在 Linux 的防火墙体系 Netfilter 下有一个独立的模块 L7 filter。从字面上看 Netfilter 是对网络数据的过滤，L7 filter 是基于数据流应用层内容的过滤。不过实际上 L7 filter 的本职工作不是对数据流进行过滤而是对数据流进行分类。它使用模式匹配算法把进入设备的数据包应用层内容与事先定义好的协议规则进行比对，如果匹配成功就说明这个数据包属于某种协议。
L7 filter 是基于数据流工作的，建立在 Netfilter connstrack 功能之上。因为一个数据流或者说一个连接的所有数据都是属于同一个应用的，所以 L7 filter 没有必要对所以的数据包进行模式匹配，而只匹配一个流的前面几个数据包 （比如 5 或 10 个数据包）。当一个流的前

面几个数据包包含了某种应用层协议的特征码时 （比如 QQ），则这个数据流被 L7 filter 识别；当前面几个数据包的内容没有包含某种应用层协议的特征码时，则 L7 filter 放弃继续做模式匹配，这个数据流也就没有办法被识别。

iptables 常用配置实例

1、阻止相应网段主机访问服务端指定端口服务
10.0.0.0/24 -- 22 端口（阻止）
iptables -t filter -A INPUT -s 10.0.0.0/24 -p tcp --dport 22 -j DROP
iptables -t filter -A INPUT -s 10.0.0.9 -p tcp --dport 22 -j DROP
iptables -t filter -A INPUT -i eth0 -s 10.0.0.9 -p tcp --dport 22 -j DROP

2、除了某个地址可以访问 22 端口之外，其余地址都不能访问
10.0.0.1 10.0.0.253　　10.0.0.9（只允许）
iptables -t filter -A INPUT -s 10.0.0.9 -p tcp --dport 22 -j ACCEPT
iptables -t filter -A INPUT -s 10.0.0.0/24 -p tcp --dport 22 -j DROP
iptables -t filter -A INPUT ! -s 10.0.0.9 -p tcp --dport 22 -j ACCEPT
！（感叹号）表示非

3、指定阻止访问多个端口服务
22--80 22,24,25
iptables -A INPUT -s 10.0.0.9 -p tcp --dport 22:80 -j DROP
　　--- 匹配连续的端口号访问
iptables -A INPUT -s 10.0.0.9 -m multiport -p tcp --dport 22,24,25 -j DROP
　　--- 匹配不连续的端口号访问，默认不支持，必须加扩展参数 multiport
　-m　--- 指定应用扩展模块参数
　multiport　--- 可以匹配多个不连续端口信息

4、通过防火墙实现禁 ping 功能
实现 ping 功能测试链路是否正常，基于 icmp 协议实现的
icmp 协议有多种类型（这里只需关注 0 和 8）：
icmp-type 8：请求类型　icmp-type 0：回复类型

情况一：实现禁止主机访问防火墙服务器（禁 ping，主机主动）

iptables -A INPUT -p icmp --icmp-type 8 -j DROP
　　--直接不允许 ping 进来
iptables -A OUTPUT -p icmp --icmp-type 0 -j DROP
　　--允许进来，但是不允许防火墙回复

情况二：实现禁止防火墙访问主机服务器（禁 ping，防火墙主动）

```
iptables -A OUTPUT -p icmp --icmp-type 8 -j DROP
            --防火墙不让 ping 请求出去
iptables -A INPUT -p icmp --icmp-type 0 -j DROP
            --防火墙可以让请求发送，但是不让回复进来

默认情况：所有 icmp 类型都禁止
iptables -A INPUT -p icmp -m icmp --icmp-type any -j DROP
iptables -A OUTPUT -p icmp -m icmp --icmp-type any -j DROP
```

5、实现防火墙状态机制控制

--根据连接追踪机制检查连接的状态

--能够增强服务器（比如 web 服务）的安全性，避免反弹式木马攻击，但是必须打开连接追踪功能，这样会消耗服务器的性能。反弹式木马其实就是被动等待肉鸡来连接的一种木马，如灰鸽子，一般防火墙对内部访问外部的数据都不会过问，肉鸡会主动连接控制端。

调整连接追踪功能所能够容纳的最大连接数量：

/proc/sys/net/nf_conntrack_max

已经追踪到并记录下的连接：

/proc/net/nf_conntrack

不同协议或连接类型追的时长：

/proc/sys/net/netfilter/

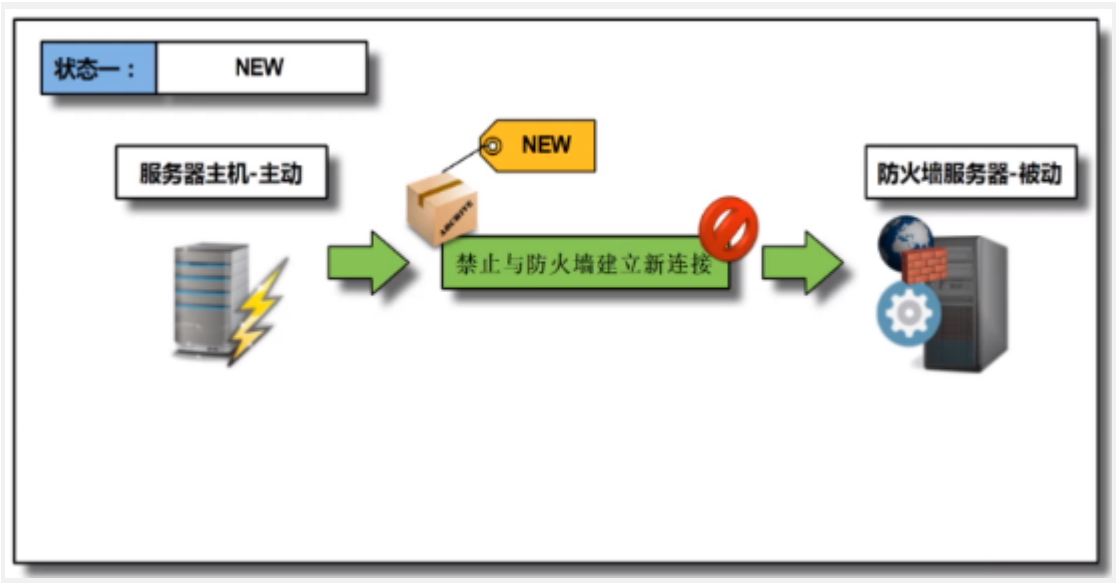
可追踪的连接状态：

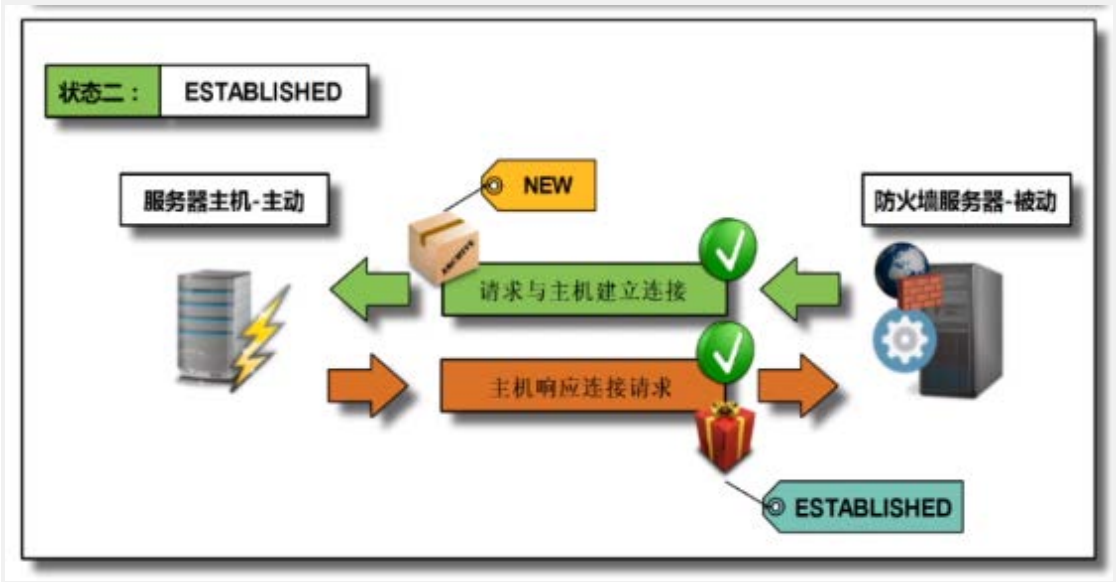
NEW：新发出的请求；连接追踪模板中不存此连接相关的信息条目，因此，将其识别为第一次发出的请求；发送数据包里面控制字段为 syn=1，发送第一次握手的数据包。

ESTABLISHED：NEW 状态之后，连接追踪模板中为其建立的条目失效之前期间内所进行的通信的状态；请求数据包发出之后，响应回来的数据包称为回复的包

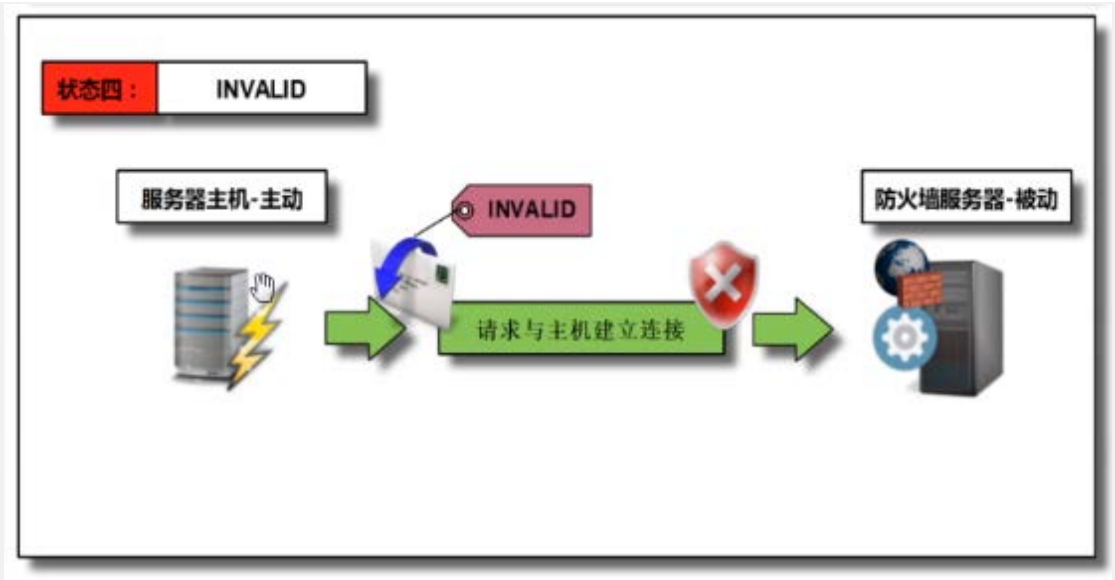
RELATED：相关的连接；基于一个连接，然后建立新的连接，如 ftp 协议的命令连接与数据连接之间的关系；

INVALIED：无法识别的连接；无效的的数据包，数据包结构不符合正常要求的


状态一：
NEW
服务器主机-主动
NEW
防火墙服务器-被动
禁止与防火墙建立新连接


状态二：
ESTABLISHED
服务器主机-主动
NEW
防火墙服务器-被动
请求与主机建立连接
主机响应连接请求
ESTABLISHED


状态三：
RELATED
NEW
服务器主机-主动
防火墙服务器-被动
请求与主机建立连接
进行新的数据包通讯
RELATED
主机响应连接请求
ESTABLISHED



例如：iptables -A INPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
--防火墙放行 FTP 主动模式的配置

6、防火墙放行 FTP 的主被动模式
开放主动模式
iptables -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -s x.x.x.x --dport 20:21 -j ACCEPT

开放被动模式方法一：
如果用 vsftpd，修改配置文件，设置最小和最大打开端口
pasv_min_port=2222
pasv_max_port=2225
iptables -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -s x.x.x.x --dport 21 -j ACCEPT
iptables -A INPUT -p tcp -s x.x.x.x --dport 2222:2225 -j ACCEPT
方法二使用连接追踪模块：
(1) 装载 ftp 追踪时的专用的模块:
/lib/modules/2.6.32-696.el6.x86_64/kernel/net/netfilter/nf_conntrack_ftp
modinfo nf_conntrack_ftp.ko
modprobe nf_conntrack_ftp
(2) 放行请求报文：
命令连接：NEW, ESTABLISHED
数据连接：RELATED, ESTABLISHED
iptables -A INPUT -d LocalIP -p tcp --dport 21 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -d LocalIP -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
(3) 放行响应报文：

ESTABLISEHD
iptables -A OUTPUT -s LocalIP -p tcp -m state --state ESTABLISHED -j ACCEPT

7、如何保存及重载规则
直接保存：规则自动保存到了/etc/sysconfig/iptables，用此命令保存的规则开机会自动生效。
iptables save
保存规则至指定文件：
iptables-save > /etc/sysconfig/iptables
从指定文件重载规则：
iptables-restore < /etc/sysconfig/iptables
保存规则：#iptables-save >/etc/iptables-script
恢复规则：#iptables-restore>/etc/iptables-script
开机自动恢复规则，把恢复命令添加到启动脚本：echo '/sbin/iptables-restore /etc/iptables-script' >>/etc/rc.d/rc.local

8、iptables 实现共享上网方法(postrouting，外网地址确定)
1）先配置内网服务器，设置网关地址为防火墙的内网地址
说明：内网服务器网关地址指定为共享上网服务器内网网卡地址

2）配置共享上网服务器（防火墙），开启共享上网服务器路由转发功能
vim /etc/sysctl.conf
net.ipv4.ip_forward = 1
sysctl -p

3）配置共享上网服务器，实现内网访问外网的 NAT 映射
iptables -t nat -A POSTROUTING -s 172.16.1.0/24 -o eth0 -j SNAT --to-source 10.0.0.7
-s 172.16.1.0/24 --- 指定将哪些内网网段进行映射转换
-o eth0 --- 指定在共享上网哪个网卡接口上做 NAT 地址转换
-j SNAT --- 将源地址进行转换变更
-j DNAT --- 将目标地址进行转换变更
--to-source ip 地址 --- 将源地址映射为什么 IP 地址
--to-destination ip 地址 --- 将目标地址映射为什么 IP 地址

如果 forward 默认 drop 策略，就需要配置 forward 链
iptables -A FORWARD -i eth1 -s 172.16.1.0/24 -j ACCEPT
iptables -A FORWARD -o eth0 -s 172.16.1.0/24 -j ACCEPT
iptables -A FORWARD -i eth0 -d 172.16.1.0/24 -j ACCEPT
iptables -A FORWARD -o eth1 -d 172.16.1.0/24 -j ACCEPT

网络数据包传输过程一定是有去有回的

9、iptables 实现共享上网方法(postrouting，外网地址不确定)

```
iptables -t nat -A POSTROUTING -s 172.16.1.0/24 -o eth0 -j MASQUERADE
                                    <-  伪装共享上网
```

说明：在企业中如何没有固定外网 IP 地址，可以采取以上伪装映射的方式进行共享上网

总结：配置映射方法

01. 指定哪些网段需要进行映射　　-s 172.16.1.0/24
02. 指定在哪做映射　　-o eth0
03. 用什么方法做映射　　-j SNAT/DNAT
04. 映射成什么地址　　--to-source　ip 地址/--to-destination ip 地址

10、iptables 实现外网 IP 的端口映射到内网 IP 的端口（prerouting）

需求：将网关的 IP 和 9000 端口映射到内网服务器的 22 端口

端口映射 10.0.0.7:9000 -->172.16.1.8:22

实现命令：

```
iptables -t nat -A PREROUTING -d 10.0.0.7  -i eth0 -p tcp --dport 9000 -j DNAT --to-destination 172.16.1.8:22
```

-d 10.0.0.8　　目标地址。
-j DNAT　　目的地址改写。

firewalld

在 RHEL7 系列中，默认使用 firewalld 作为防火墙，iptables 默认是关闭的（stop），要使用 firewalld 建议先关闭并禁用老版本的 iptables，ip6tables 和 ebtables。

```
systemctl mask iptables
systemctl mask ip6tables
systemctl mask ebtables
systemctl enable firewalld
```

区域概念

在 firewalld 中引入了区域的概念，可以简单理解为一些配置好策略的模板，直接应用即可，如果是学网络的，应该很能理解，比如华为的防火墙，华三的防火墙这些，都是有这种概念的，这里就不详细解释什么是区域了。

zone	default configuration
trusted	许任何流入数据包和流出数据包

home	任何流入的数据包都将拒绝,但允许数据包流出和预定义服务（ssh、mdns、ipp-client、sbmclient、dhcpv6-client）的数据包可以流入
internal	同上
work	任何流入的数据包都将拒绝,但允许数据包流出和预定义服务（ssh、ipp-client、dhcpv6-client）的数据包可以流入
public	任何流入的数据包都将拒绝,但允许数据包流出和预定义服务（ssh、dhcpv6-client）的数据包可以流入.新添加的网卡默认绑定到该区
external	任何流入的数据包都将拒绝,但允许数据包流出和预定义服务（ssh）的数据包可以流入.所有从该区出去的数据包都将映射成该区所绑定的网卡的 IP。
dmz	任何流入的数据包都将拒绝,但允许数据包流出和预定义服务（ssh）的数据包可以流入.
block	任何流入的数据包都将拒绝,但允许数据包流出
drop	任何流入网络数据包都将被丢弃,不作出任何回应.只允许数据包出去.

zone	means
trusted	可接受所有的网络连接。 指定其中一个区域为默认区域是可行的。当接口连接加入了 NetworkManager，它们就被分配为默认区域。安装时，firewalld 里的默认区域被设定为公共区域
home	用于家庭网络。您可以基本信任网络内的其他计算机不会危害您的计算机。仅仅接收经过选择的连接。
internal	用于内部网络。您可以基本上信任网络内的其他计算机不会威胁您的计算机。仅仅接受经过选择的连接。
work	用于工作区。您可以基本相信网络内的其他电脑不会危害您的电脑。仅仅接收经过选择的连接。
public	在公共区域内使用，不能相信网络内的其他计算机不会对您的计算机造成危害，只能接收经过选取的连接。
external	特别是为路由器启用了伪装功能的外部网。您不能信任来自网络的其他计算，不能相信它们不会对您的计算机造成危害，只能接收经过选择的连接。
dmz	用于您的非军事区内的电脑，此区域内可公开访问，可以有限地进入您的内部网络，仅仅接收经过选择的连接。
block	任何接收的网络连接都被 IPv4 的 icmp-host-prohibited 信息和 IPv6 的 icmp6-adm-prohibited 信息所拒绝。
drop	任何接收的网络数据包都被丢弃，没有任何回复。仅能有发送出去的网络连接。

firewalld 命令基础使用

只有 firewalld 服务启动了，才能使用相关工具： firewall-config(图形界面), firewall-cmd, 这里主要介绍命令行方式的。

firewalld 有规则两种状态
运行时（runtime)：修改规则马上生效,但是临时生效
持久配置（permanent）： 修改后需要重载才会生效
firewall-cmd --permanent RULE
firewall-cmd --reload
注意：一旦使用了 permanent，配置完成后一定要 reload，否则只能待防火墙重启后这些配置才能生效。

配置文件
/etc/firewalld/{services,zones}/*.xml 优先级最高，permanent 模式生效的策略会放到这里
/lib/firewalld/{services,zones}/*.xml 优先级要低些，是一些默认配置，可以当做模板使用

以下 options 是可以组合使用的，具体见 man firewall-cmd 或者 firewall-cmd --help，里面有详细的说明，此处只列出一些常用的 option。
firewall-cmd
--permanent --配置写入到配置文件，否则临时马上生效
--reload --重载配置文件，永久生效
--zone= --指定区域
--get-default-zones --获取默认区域
--set-default-zone= --设置默认区域
--get-zones --获取所有可用的区域
--get-active-zones --获取当前激活（活跃）的区域
--add-source= --添加地址，可以是主机或网段，遵循当前区域的 target
--remove-source --移除地址，可以是主机或网段，遵循当前区域的 target
--add-service= --添加服务，遵循当前区域的 target
--remove-service= --移除服务，遵循当前区域的 target
--list-services --显示指定区域内允许访问的所有服务
--add-port= --添加端口，遵循当前区域的 target
--remove-port= --移除端口，遵循当前区域的 target
--list-ports 显示指定区域内允许访问的所有端口号
--add-interface= --zone= --添加网卡到指定区域
--change-interface= --new-zone-name --改变网卡到指定区域
--list-all --列出激活使用的区域的配置
--list-all-zones --列出所有区域的配置
--get-zone-of-interface= --获取指定接口所在的区域
--list-icmp-blocks --显示指定区域内拒绝访问的所有 ICMP 类型
--add-icmp-block= --为指定区域设置拒绝访问的某项 ICMP 类型

```
    --remove-icmp-block=     --移除指定区域内拒绝访问某项的 ICMP 类型
        常用的 ICMP 类型有
            #echo-request：类型 0，icmp 请求报文
            #echo-reply：类型 8，icmp 响应回复报文


    --list-protocols         --列出在指定区域中允许通过的协议
    --add-protocol=          --在指定区域中添加允许通过的协议
    --remove-protocol=       --移除在指定区域中的某项协议
    --get-target             --获取区域中的默认 target
    --set-target=            --设置区域的 target
查看默认区域
firewall-cmd --get-default-zone
查看所有可以使用的区域
firewall-cmd --get-zones

修改当前的默认区为 work
firewall-cmd --set-default-zone=work
在 work 区添加 http 服务并允许别人访问
firewall-cmd --add-service=http --zone=work

在 public 区绑定了该地址范围，只有该范围的 IP 的数据包都会路由到该区，由该区的规则进行匹配决定是否放行
firewall-cmd --add-source=172.25.0.10/32 --zone=public
查看活跃的区
firewall-cmd --get-active-zones
把接口绑定到 public 区
--change-interface= --new-zone-name
--add-interface= --zone=
查看网卡绑定在了哪个区域
firewall-cmd --get-zone-of-interface=IFACE

查看区域配置信息
firewall-cmd --list-all --zone=work
firewall-cmd --list-all

查看所有服务
firewall-cmd --get-services
```

直接规则

```
Direct Options
```

```
  --direct                    --指定将要使用直接规则
  --get-all-chains            --获取所有的链
  --get-chains {ipv4|ipv6|eb} <table>     --获取表中的链
  --add-chain {ipv4|ipv6|eb} <table> <chain>      --添加链到表中（自定义的）
  --remove-chain {ipv4|ipv6|eb} <table> <chain> --移除表中的某条链
  --query-chain {ipv4|ipv6|eb} <table> <chain>    --返回链是否已被添加到表
  --get-all-rules             --获取所有的策略规则
  --get-rules {ipv4|ipv6|eb} <table> <chain>      --获取指定表中指定链中的规则
  --add-rule {ipv4|ipv6|eb} <table> <chain> <priority> <arg>...       --在指定表中的指定链添加规则
  --remove-rule {ipv4|ipv6|eb} <table> <chain> <priority> <arg>...  --优先从指定表中的指定链删除规则条目
  --remove-rules {ipv4|ipv6|eb} <table> <chain>       --从表中删除规则条目
  --query-rule {ipv4|ipv6|eb} <table> <chain> <priority> <arg>...    --返回是否优先的规则已被添加到链表中
  --passthrough {ipv4|ipv6|eb} <arg>...   --直接传递 iptables 的命令
--get-all-passthroughs          --获取所有直接传递的规则
  --get-passthroughs {ipv4|ipv6|eb} <arg>...        --获取跟踪直通规则
  --add-passthrough {ipv4|ipv6|eb} <arg>...         --添加一个直接规则
  --remove-passthrough {ipv4|ipv6|eb} <arg>...    --移除一个直接规则
  --query-passthrough {ipv4|ipv6|eb} <arg>...       --查询直接规则是否添加
```

除非将直接规则显式插入到由 fircwa1ld 管理的区域， 否则将首先解析直接规则， 然后才会解析任何 fircwalld 规则，即直接规则优先于 firewalld 规则。

示例 1：添加一些直接规则以将某个 IP 范围列入黑名单的简短示例：
直接规则主要用于使服务和应用程序能够增加规则。传递的参数 与 iptables, ip6tables 以及 ebtables 一致。

所有来自 192.168.0.0/24 网络 IP，单个 IP 每分钟最高连接并发是 1，超过并发的连接都丢弃

```
firewall-cmd --direct --permanent --add-chain ipv4 raw blacklist
firewall-cmd --direct --permanent --add-rule ipv4 raw PREROUTING O -s 192. 168, 0. 0/24 -j blacklist
# 注意，这里-j 在前面 iptables 时我们说过，是可以指定自定义链这一 target 的
firewall-cmd --direct --permanent --add-rule ipv4 raw blacklist O -m limit --limit 1/min -j LOG --log-prefix "blacklisted "
firewall-cmd --direct --permanent --add-rule ipv4 raw blacklist 1 -j DROP

firewall-cmd --reload
firewall-cmd --direct --get-all-rules
      ipv4 raw PREROUTING 0 -s 192.168.0.0/24 -j blacklist
      ipv4 raw blacklist 0 -m limit --limit 1/min -j LOG --log-prefix blacklisted
```

```
    ipv4 raw blacklist 1 -j DROP

实验完毕，删除规则
firewall-cmd --permanent --direct --remove-rules ipv4 raw blacklist
firewall-cmd --permanent --direct --remove-rules ipv4 raw PREROUTING

firewall-cmd --direct --remove-rules ipv4 raw blacklist
firewall-cmd --direct --remove-rules ipv4 raw PREROUTING
firewall-cmd --direct --remove-chain ipv4 raw blacklist

firewall-cmd --reload
```

```
示例 2：利用 firewalld 直接规则来做 SNAT 和 MASQUERADE 实现内网共享上网
firewall-cmd --permanent --direct --passthrough ipv4 -t nat -I POSTROUTING -o eth1 -j MASQUERADE -s 172.25.0.0/24

firewall-cmd --permanent --direct --passthrough ipv4 -t nat -I POSTROUTING -o eth1 -j SNAT -s 172.25.0.0/24 --to-source <外网 IP>

firewall-cmd --direct --get-all-passthroughs
```

firewalld 防火墙富规则

```
富规则一般为了精确控制，用得比较多，所以这里将重点讲解
```

富规则常用的 options

```
firewall-cmd
    --list-rich-rules                   --列出富规则
    --add-rich-rule=<rule>              --使用富规则语言添加富规则
    --remove-rich-rule=<rule>           --移除富规则
    --query-rich-rule=<rule>            --查询某条富规则是否存在
    --list-rich-rules                   --列出指定区域中的所有富规则
    --list-all 和 --list-all-zones      --也能列出存在的富规则
```

富规则 language 语法

规则的几乎每个单一元素都能够以 option=value 形式来采用附加参数，参考 firewalld.richlanguage

```
rule
[source]
[destination]
service|port|protocol|icmp-block|masquerade|forward-port
[log]
[audit]
[accept|reject|drop]


rule [family="ipv4|ipv6"]
source address="address[/mask]" [invert="True"]
destination address="address[/mask]" invert="True"
service name="service name"
port port="port value" protocol="tcp|udp"
protocol value="protocol value"
forward-port port="port value" protocol="tcp|udp" to-port="port value
" to-addr="address"
log [prefix="prefix text"] [level="log level"] [limit value="rate/duration"]
accept | reject [type="reject type"] | drop
```

一些常用配置规则

规则排序

一旦向某个区域（一般是指防火墙）中添加了多个规则， 规则的排序便会在很大程度上影响防火墙的行为。
对于所有区域， 区域内规则的基本排序是相同的。
1.为该区域设置的任何端口转发和伪装规则。
2.为该区域设置的任何记录规则。
3.为该区域设置的任何允许规则。
4.为该区域设置的任何拒绝规则。
在所有情况下， 第一个匹配项都将胜出。如果区域中的任何规则与包均不匹配 ， 那

么通常会拒绝该包 但是区域可能具有不同默认值；例如，可信区域将接受任何不匹配的包。此外，在匹配某个记录规则后，将继续正常处理包。

直接规则是个例外。对于大部分直接规则，将首先进行解析，然后再由 firewalld 进行任何其他处理 ，但是直接规则语法允许管理员在任何区域中的任何位置插入任何规则。

测试和调试

为了便于测试和调试 几乎所有规则都可以与超时一起添加到运行时配置。
当包含超时的规则到防火墙时，计时器便针对该规则开始倒计时。
一旦规则的计时器达到零秒，便从运行时配置中删除该规则。
在使用远程防火墙时，使用超时会是一种极其有用的工具，特别是在测试更复杂的规则集时。
如果规则有效，则管理员可以再次添加该规则，但使用--permanent 选项（或者至少不包含超时）。
如果规则没有按预期运行，甚至可能将管理员锁定而使其无法进入系统，那么规则将被自动删除，
以允许管理员继续其工作。通过在启用规则的 firewall-cmd 的结尾添加--timeout=<timeinseconds>
即可向运行时规则中添加超时。

使用富规则进行日志记录

调试或监控防火墙时，记录已接受或已拒绝的连接很有用。
firewalld 可以通过两种方法来实现此目的记录到 syslog，或者将消息发送到由 auditd 管理的内核 audit 子系统。
在这两种情况下，记录可能会受到速率限制。
速率限制确保系统日志文件填充消息的速率不会使系统无法跟上或者填充其所有磁盘空间 。

使用富规则记录到 syslog 的基本语法为：
log [prefix="<PREFIX TEXT>" [level=<LOGLEVEL>] [limit value="<RATE/DURATION>"]
其中：
<LOGLEVEL> 可以是 emerg 、 alert 、 crit 、 error 、 warning 、 notice 、 info 或 debug 之一。
<RATE/DURATION>" 可以是 s(表示秒）、m(表示分钟）、h(表示小时）或 d(表示天）之一。
例如：limit value=3/m 会将日志消息限制为每分钟最多三条。
limit value = 3/m 这里是有 BUG 的，常常时间控制会不精准。

批量加端口

port port=4000-5234 用这样的形式
firewall-cmd --permanent --zone=vnc --add-rich-rule='rule family=ipv4 source address=192.168.1.0/24 port port=7900-7905 protocol=tcp accept'

用于记录到审计子系统的基本语法为：
audit [limit value="<RATE/DURATION>"]
速率限制的配置方式与 syslog 记录相同。

使用富规则进行记录的某些示例：
1.接受从 work 区域到 SSH 的新连接， 以 notice 级别且每分钟最多三条消息的方式将新连接记录 到 syslog。
firewall-cmd --permanent --zone=work --add-rich-rule='rule service name="ssh" log prefix="ssh" level="notice" limit va1ue="3/m" accept
2.在接下来五分钟内， 将拒绝从默认区域中子网 2001: db8: : /64 到 DNS 的新 IPv6 连接，并且拒绝的连接将记录到 audit 系统，且每小时最多一条消息。
firewall-crud --add-rich-rule='rule family=ipv6 source address="2001 :db8: :/64" service name=" dns" audit limit va1ue=" 1/h" reject'--timeout=300

一些配置示例

firewall-cmd --permanent --zone=public --add-rich-rule='rule family=ipv4 source address=192.168.0.11/32 reject'
firewall-cmd --permanent --zone=public --add-rich-rule='rule service name=ftp limit value=1/m accept'
firewall-cmd --permanent --zone=public --add-rich-rule='rule protocol value=esp drop'
firewall-cmd --permanent --add-rich-rule='rule family=ipv4 source address=172.17.10.0/24 service name=ssh reject'
firewall-cmd --permanent --zone=public --add-rich-rule='rule family=ipv4 source address=192.168.0.0/24 port port=7900-7905 protocol=tcp accept'

针对 ssh 链接记录至日志中，每分钟 3 次
firewall-cmd --permanent --zone=work --add-rich-rule='rule service name=ssh log prefix="ssh " level=notice limit value="3/m" accept'

用于调试，规则在 300 秒后失效，防止规则设定错误导致网络连接断开
firewall-cmd --add-rich-rule='rule family=ipv4 source address=192.168.0.11/32 service name=ssh reject' --timeout=300

添加富规则，只允许 172.25.0.10/32 访问，并且记录日志，日志级别为 notice,日志前缀为 "NEW HTTP ",限制每秒最多 3 个并发 ，要求持久化生效

```
firewall-cmd --permanent --add-rich-rule='rule family=ipv4 source
address=172.25.0.10/32 service name=http log level=notice prefix="NEW HTTP " limit
value="3/s" accept'

firewall-cmd --reload
```

firewalld 的 NAT 和端口转发

1）网络地址转换（NAT）
firewalld 支持两种类型的网络地址转换(NAT) 伪装和端口转发。可以在基本级别使用常规 frewall-cmd 规则来同时配置这两者，更高级的转发配置可以使用富规则来完成。
这两种形式的 NAT 会在发送包之前修改包的某些方面如源或目标。
2）伪装（注意：伪装只能和 ipv4 一起用，ipv6 不行）
通过伪装系统会将并非直接寻址到自身的包转发到指定接收方同时将通过的包的源地址更改为其自己的公共 TP 地址。防火墙对这些传入的包应答时会将目标地址修改为原始主机的地址并发送包。这通常在网络边缘上使用以便为内部网络提供 Internet 访问。伪装是一种形式的网络地址转换（NAT）
3）配置伪装
要使用常规的 firewall-cmd 命令为区域配置伪装，使用下列语法：
firewall-cmd --permanent --zone=<ZONE> --add-masquerade
这将伪装满足以下条件的任何包从该区域的源（接口及子网）中定义的客户端发送到防火墙且未寻址到防火墙自身的包，即该区域源的数据包的目标 IP 不是防火墙自身的 IP，都将映射成防火墙的 IP。
要在更大程度上控制要进行伪装的客户端,还可以使用富规则。
firewall-cmd --permanent --zone=<ZONE> --add-rich-rule='rule family=ipv4 source address=192.168.0.0/24 masquerade'

4）端口转发
另一种形式的 NAT 是端口转发。
通过端口转发指向单个端口的流量将转发到相同计算机上的不同端口或者转发到不同计算机上的端口。

此机制通常用于将某个服务器隐藏在另一个计算机后面，或者用于在备用端口上提供对服务的访问权限。

重要：
当端口转发配置为将包转发到不同计算机时，从该计算机的任何回复通常将直接从该计算机发送到原始客户端。这将在大部分配置上导致无效连接，因此转发到的计算机必须通过执行端口转发的防火墙来进行伪装。常见配置是将端口从防火墙计算机转发到已在防火墙后面伪装的计算机，即这种通过目标端口转发的方式需要开启伪装（masquerade）

要使用常规 fircwall-cmd 命令配置端口转发，可以使用 firewalld 提供的端口转发语法：
firewall-cmd --permanent --zone=<ZONE> --add-forward-port=port=<PORTNUMBER>:proto=<PROTOCOL>[:toport=<PORTNUMBER>][:toaddr=<IP ADDR>]

toport=和 toaddr 两部分均可选，但需要至少指定这两者之一 。
例如， 对于来自 public 区域的客户端，以下命令会将防火墙上通过端口 513/TCP 传入的连接转发到 IP 地址为 192.168.0.254 的计算机上的端口 132/TCP
frewall-cmd --permanent --zone=public --add-forward-port=port=513:proto=tcp:toport=132:toaddr=192.168.0.254

要在更大程度上控制端口转发规则，可以将以下语法与富规则配合使用来实现端口转发，这也是我们推荐使用的方式
forward-port port=<PORTNUM> protocol=tcp|udp [to-port=<PORTNUM>] [to-addr=<ADDRESS>]
以下示例使用富规则将来自 work 区域中 192.168.0.0/4 且传入到端口 80/TCP 的流星转发到防火墙计算机自身上面的和端口 8080/TCP:
firewall-cmd --permanent --zone=work --add-rich-rule='rule family=ipv4 source address=l92.168.0.0/24 forward-port port=80 protocol=tcp to-port=8080'

其实这里是指传统的目标地址映射，实现外网访问内网资源
firewall-cmd --zone=external --add-masquerade
firewall-cmd --permanent --zone=public --add-forward-port=port=513:proto=tcp:toport=132:toaddr=192.168.0.254

firewall-cmd --permanent --zone=public --add-rich-rule='rule family=ipv4 source address=192.168.0.0/24 forward-port port=80 protocol=tcp to-port=8080'

例子：
把来自 172.25.0.10/32 并且访问的端口为 tcp 的 443 端口的数据包进行端口转发，转发到本机的 tcp 的 22 端口

firewall-cmd --permanent --add-rich-rule='rule family=ipv4 source address=172.25.0.10/32 forward-port port=443 protocol=tcp to-port=22'
firewall-cmd --reload

本地端口转发：
firewall-cmd --add-rich-rule='rule family=ipv4 source address=172.25.0.10/32 forward-port port=80 to-port=22 protocol=tcp'

目标地址转发：
firewall-cmd --add-rich-rule='rule family=ipv4 source address=172.25.0.10/32 forward-

```
port port=80 to-port=80 protocol=tcp to-addr=172.25.0.254'
```

```
一键断网
firewall-cmd --panic-off
firewall-cmd --panic-on

重新会加载内核中的防火墙模块和配置文件
firewall-cmd --complete-reload

这里值得注意的是如果主机开启了 SELinux，需要对对于的端口打上标签（SELinux context）
semanage port -l
semanage port -a -t port_label -p tcp|udp PORTNUMBER
semanage port -a -t gopher_port_t -p tcp 71

semanage 命令如果是最小化安装，可能没有这个工具，所以需要自己安装，对应的包如下
rpm -qa | grep semanage
libsemanage-2.5-8.el7.x86_64
libsemanage-python-2.5-8.el7.x86_64
```

firewalld 自定义区域和服务

自定义的区域

```
参照/lib/firewalld/zones/public.xml 文件，在/lib/firewalld/zones/下面新建你需要的区域名以.xml 结尾，内容格式参照 public.xml，trusted.xml 等文件即可。
```

Internal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Internal</short>
  <description>For use on internal networks. You mostly trust the other computers on the networks
to not harm your computer. Only selected incoming connections are accepted.</description>
  <service name="ssh"/>
  <service name="mdns"/>
  <service name="samba-client"/>
```

```
  <service name="dhcpv6-client"/>
</zone>
```

自定义服务

自定义服务可以让防火墙配置更为简便，能够直接使用服务名来进行策略配置
参照/usr/lib/firewalld/services/ssh.xml，在/usr/lib/firewalld/services/下面新建你需要的服务名以.xml 结尾，内容格式参照 ssh.xml，samba.xml 文件中的内容即可。

samba.xml

```
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>Samba</short>
  <description>This option allows you to access and participate in Windows file and printer sharin
g networks. You need the samba package installed for this option to be
useful.</description>
  <port protocol="udp" port="137"/>
  <port protocol="udp" port="138"/>
  <port protocol="tcp" port="139"/>
  <port protocol="tcp" port="445"/>
  <module name="nf_conntrack_netbios_ns"/>
</service>
```

firewalld 的一些实用操作

```
systemctl stop {ip,eb,ip6}tables  #一般默认在 RHEL7 中是没有开启的
systemctl mask {ip,eb,ip6}tables
systemctl start firewalld
systemctl status firewalld 可以看到 firewalld 启动了
配置文件
/etc/firewalld/{services,zones}/*.xml  优先级最高，permanent 模式生效的策略会放到这里
/lib/firewalld/{services,zones}/*.xml    优先级要低些，是一些默认配置，可以当做模板使用

firewall-cmd --permanent
      --使用该 option 会将配置写入到/etc/firewalld/{services,zones}/*.xml 对应的文件中，但是必须使用 firewall-cmd --reload 来重载配置才能生效，如果不使用--permanent，那么配置将会马上生效，但是在重启 firewalld 和重载（reload） firewalld 后不会生效，所以是临时的。
```

查看防火墙的运行状态
firewall-cmd --state
systemctl status firewalld
查看防火墙的配置情况
firewall-cmd --list-all

应急命令（一键断网）
firewall-cmd --panic-on　　#拒绝所有流量，远程连接会立即断开，只有本地能登陆
firewall-cmd --panic-off　　#取消应急模式，但需要重启 firewalld 后才可以远程 ssh
firewall-cmd --query-panic　　#查看是否为应急模式

添加服务和端口以及协议
firewall-cmd --add-service=<service name> #添加服务
firewall-cmd --remove-service=<service name> #移除服务

firewall-cmd --add-port=<port>/<protocol>　　#添加端口/协议（TCP/UDP）
firewall-cmd --remove-port=<port>/<protocol>　#移除端口/协议（TCP/UDP）
firewall-cmd --list-ports #查看开放的端口

firewall-cmd --add-protocol=<protocol>　　#允许协议（例：icmp，即允许 ping)
firewall-cmd --remove-protocol=<protocol> #取消协议
firewall-cmd --list-protocols　　#查看允许的协议

指定某 IP 或网段的流量通过
firewall-cmd --add-rich-rule="rule family="ipv4" source address="<ip/netmask>" accept"

firewall-cmd --add-rich-rule="rule family="ipv4" source address="192.168.2.1" accept"
　　#表示允许来自 192.168.2.1 的所有流量

指定 IP 时指定协议
firewall-cmd --add-rich-rule="rule family="ipv4" source address="<ip>" protocol value="<protocol>" accept"

firewall-cmd --add-rich-rule="rule family="ipv4" source address="192.168.2.208" protocol value="icmp" accept"
　　#允许 192.168.2.208 主机的 icmp 协议，即允许 192.168.2.208 主机 ping

允许指定 ip 访问指定服务
firewall-cmd --add-rich-rule="rule family="ipv4" source address="<ip>" service name="<service name>" accept"

firewall-cmd --add-rich-rule="rule family="ipv4" source address="192.168.2.208" service

name="ssh" accept"
#允许 192.168.2.208 主机访问 ssh 服务

允许指定 ip 访问指定端口
firewall-cmd --add-rich-rule="rule family="ipv4" source address="<ip>" port protocol="<port protocol>" port="<port>" accept"

firewall-cmd --add-rich-rule="rule family="ipv4" source address="192.168.2.1" port protocol="tcp" port="22" accept"
#允许 192.168.2.1 主机访问 22 端口

禁止 IP 和网段
firewall-cmd --zone=drop --add-rich-rule="rule family="ipv4" source address="192.168.2.0/24" port protocol="tcp" port="22" reject"
#禁止 192.168.2.0/24 网段的主机访问 22 端口。

本地端口转发：
firewall-cmd --add-rich-rule='rule family=ipv4 source address=172.25.0.10/32 forward-port port=80 to-port=22 protocol=tcp accept'

目标地址转发：
firewall-cmd --permanent --add-masquerade
firewall-cmd --add-rich-rule='rule family=ipv4 source address=172.25.0.10/32 forward-port port=80 to-port=80 protocol=tcp to-addr=172.25.0.254 accept'

要使用常规的 firewall-cmd 命令为区域配置伪装，使用下列语法：
firewall-cmd --permanent --zone=<ZONE> --add-masquerade
这将伪装满足以下条件的任何包从该区域的源（接口及子网）中定义的客户端发送到防火墙且未寻址到防火墙自身的包，即该区域源的数据包的目标 IP 不是防火墙自身的 IP，都将映射成防火墙的 IP。
要在更大程度上控制要进行伪装的客户端,还可以使用富规则。
firewall-cmd --permanent --zone=<ZONE> --add-rich-rule='rule family=ipv4 source address=192.168.0.0/24 masquerade'