**iptables** 是 netfilter 的控制器，它是 Linux 内核的包过滤框架。 默认情况下，iptables 包含在大多数 Linux 发行版中。

（<https://www.linode.com/docs/security/firewalls/control-network-traffic-with-iptables>）

**firewallD** 是可用于 CentOS/Fedora 系列发行版的 iptables 控制器。

（<https://www.linode.com/docs/security/firewalls/introduction-to-firewalld-on-centos>）

**UFW** 为 Debian 和 Ubuntu 提供了一个 iptables 前端。

（<https://www.linode.com/docs/security/firewalls/configure-firewall-with-ufw>）

**在 Ubuntu 中用 UFW 配置防火墙**

<https://linux.cn/article-8087-1.html>

UFW，即简单防火墙uncomplicated firewall，是一个 Arch Linux、Debian 或 Ubuntu 中管理防火墙规则的前端。 UFW 通过命令行使用（尽管它有可用的 GUI），它的目的是使防火墙配置简单（即不复杂uncomplicated）。

**开始之前**

1、 熟悉我们的入门指南，并完成设置服务器主机名和时区的步骤。

2、 本指南将尽可能使用 sudo。 在完成保护你的服务器（<https://linux.cn/article-8076-1.html>）指南的章节，创建一个标准用户帐户，强化 SSH 访问和移除不必要的网络服务。 但不要跟着创建防火墙部分 - 本指南是介绍使用 UFW 的，它对于 iptables 而言是另外一种控制防火墙的方法。

3、 更新系统

Arch Linux

sudo pacman -Syu

Debian / Ubuntu

sudo apt-get update && sudo apt-get upgrade

**安装 UFW**

UFW 默认包含在 Ubuntu 中，但在 Arch 和 Debian 中需要安装。 Debian 将自动启用 UFW 的 systemd 单元，并使其在重新启动时启动，但 Arch 不会。 这与告诉 UFW 启用防火墙规则不同，因为使用 systemd 或者 upstart 启用 UFW 仅仅是告知 init 系统打开 UFW 守护程序。

默认情况下，UFW 的规则集为空，因此即使守护程序正在运行，也不会强制执行任何防火墙规则。 强制执行防火墙规则集的部分在下面。

**Arch Linux**

1、 安装 UFW：

sudo pacman -S ufw

2、 启动并启用 UFW 的 systemd 单元：

sudo systemctl start ufw

sudo systemctl enable ufw

**Debian / Ubuntu**

1、 安装 UFW

sudo apt-get install ufw

**使用 UFW 管理防火墙规则**

**设置默认规则**

大多数系统只需要打开少量的端口接受传入连接，并且关闭所有剩余的端口。 从一个简单的规则基础开始，ufw default命令可以用于设置对传入和传出连接的默认响应动作。 要拒绝所有传入并允许所有传出连接，那么运行：

sudo ufw default allow outgoing

sudo ufw default deny incoming

ufw default 也允许使用 reject 参数。

警告：

除非明确设置允许规则，否则配置默认 deny 或 reject 规则会锁定你的服务器。确保在应用默认 deny 或 reject 规则之前，已按照下面的部分配置了 SSH 和其他关键服务的允许规则。

**添加规则**

可以有两种方式添加规则：用端口号或者服务名表示。

要允许 SSH 的 22 端口的传入和传出连接，你可以运行：

sudo ufw allow ssh

你也可以运行：

sudo ufw allow 22

相似的，要在特定端口（比如 111）上 deny 流量，你需要运行：

sudo ufw deny 111

为了更好地调整你的规则，你也可以允许基于 TCP 或者 UDP 的包。下面例子会允许 80 端口的 TCP 包：

sudo ufw allow 80/tcp

sudo ufw allow http/tcp

这个会允许 1725 端口上的 UDP 包：

sudo ufw allow 1725/udp

**高级规则**

除了基于端口的允许或阻止，UFW 还允许您按照 IP 地址、子网和 IP 地址/子网/端口的组合来允许/阻止。

允许从一个 IP 地址连接：

sudo ufw allow from 123.45.67.89

允许特定子网的连接：

sudo ufw allow from 123.45.67.89/24

允许特定 IP/ 端口的组合：

sudo ufw allow from 123.45.67.89 to any port 22 proto tcp

proto tcp 可以删除或者根据你的需求改成 proto udp，所有例子的 allow 都可以根据需要变成 deny。

**删除规则**

要删除一条规则，在规则的前面加上 delete。如果你希望不再允许 HTTP 流量，你可以运行：

sudo ufw delete allow 80

删除规则同样可以使用服务名。

**编辑 UFW 的配置文件**

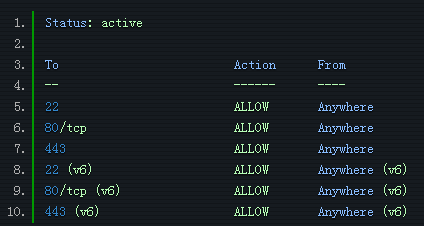
虽然可以通过命令行添加简单的规则，但仍有可能需要添加或删除更高级或特定的规则。 在运行通过终端输入的规则之前，UFW 将运行一个文件 before.rules，它允许回环接口、ping 和 DHCP 等服务。要添加或改变这些规则，编辑 /etc/ufw/before.rules 这个文件。 同一目录中的 before6.rules 文件用于 IPv6 。

还存在一个 after.rule 和 after6.rule 文件，用于添加在 UFW 运行你通过命令行输入的规则之后需要添加的任何规则。

还有一个配置文件位于 /etc/default/ufw。 从此处可以禁用或启用 IPv6，可以设置默认规则，并可以设置 UFW 以管理内置防火墙链。

**UFW 状态**

你可以在任何时候使用命令：sudo ufw status 查看 UFW 的状态。这会显示所有规则列表，以及 UFW 是否处于激活状态：



**启用防火墙**

随着你选择规则完成，你初始运行 ufw status 可能会输出 Status: inactive。 启用 UFW 并强制执行防火墙规则：

sudo ufw enable

相似地，禁用 UFW 规则：

sudo ufw disable

UFW 会继续运行，并且在下次启动时会再次启动。

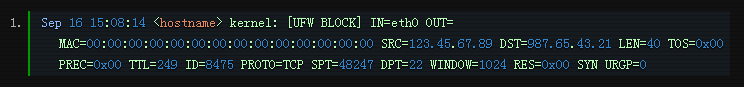
**日志记录**

你可以用下面的命令启动日志记录：

sudo ufw logging on

可以通过运行 sudo ufw logging low|medium|high 设计日志级别，可以选择 low、 medium 或者 high。默认级别是 low。

常规日志类似于下面这样，位于 /var/logs/ufw：



前面的值列出了你的服务器的日期、时间、主机名。剩下的重要信息包括：

* [UFW BLOCK]：这是记录事件的描述开始的位置。在此例中，它表示阻止了连接。
* IN：如果它包含一个值，那么代表该事件是传入事件
* OUT：如果它包含一个值，那么代表事件是传出事件
* MAC：目的地和源 MAC 地址的组合
* SRC：包源的 IP
* DST：包目的地的 IP
* LEN：数据包长度
* TTL：数据包 TTL，或称为 time to live。 在找到目的地之前，它将在路由器之间跳跃，直到它过期。
* PROTO：数据包的协议
* SPT：包的源端口
* DPT：包的目标端口
* WINDOW：发送方可以接收的数据包的大小
* SYN URGP：指示是否需要三次握手。 0 表示不需要。

**Iptables与Firewalld防火墙**

<http://www.linuxprobe.com/chapter-08.html>

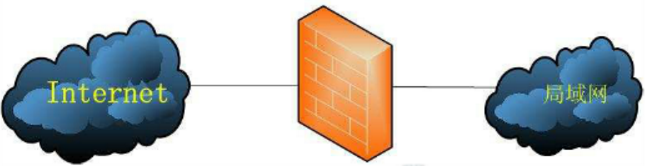
红帽RHEL7系统已经用firewalld服务替代了iptables服务，新的防火墙管理命令firewall-cmd与图形化工具firewall-config。

本章节基于数十个防火墙需求，使用规则策略完整演示对数据包的过滤、SNAT/SDAT技术、端口转发以及负载均衡等实验。

不光光学习iptables命令与firewalld服务，还新增了Tcp\_wrappers防火墙服务小节，简单配置即可保证系统与服务的安全。

**1 了解防火墙管理工具**

防火墙虽有软件或硬件之分但主要功能还是依据策略对外部请求进行过滤，成为公网与内网之间的保护屏障，防火墙会监控每一个数据包并判断是否有相应的匹配策略规则，直到满足其中一条策略规则为止，而防火墙规则策略可以是基于来源地址、请求动作或协议来定制的，最终仅让合法的用户请求流入到内网中，其余的均被丢弃。



在红帽RHEL7系统中firewalld服务取代了iptables服务，但依然可以使用iptables命令来管理内核的netfilter。这对于接触Linux系统比较早或学习过红帽RHEL6系统的读者来讲，突然接触firewalld服务会比较抵触，可能会觉得新增Firewalld服务是一次不小的改变，其实这样讲也是有道理的。但其实Iptables服务与Firewalld服务都不是真正的防火墙，它们都只是用来定义防火墙规则功能的“防火墙管理工具”，将定义好的规则交由内核中的netfilter即网络过滤器来读取，从而真正实现防火墙功能，所以其实在配置规则的思路上是完全一致的，而我会在本章中将iptables命令与firewalld服务的使用方法都教授给你们，坦白讲日常工作无论用那种都是可行的。

**2 Iptables命令**

iptables命令用于创建数据过滤与NAT规则，主流的Linux系统都会默认启用iptables命令，但其参数较多且规则策略相对比较复杂。

**2.1 规则链与策略**

在iptables命令中设置数据过滤或处理数据包的策略叫做规则，将多个规则合成一个链。

举例来说:小区门卫有两条的规则，将这两个规则可以合成一个规则链：

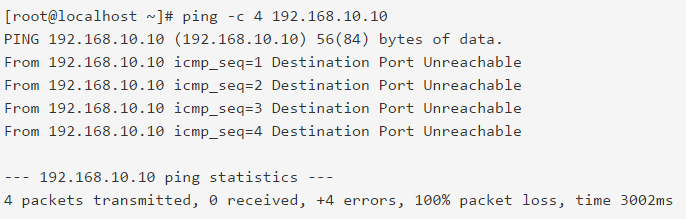
* 遇到外来车辆需要登记。
* 严禁快递小哥进入社区。

但是光有策略还不能保证社区的安全，我们需要告诉门卫（iptables）这个策略（规则链）是作用于哪里的，并赋予安保人员可能的操作有这些，如：“允许”，“登记”，“拒绝”，“不理他”，对应到iptables命令中则常见的控制类型有：

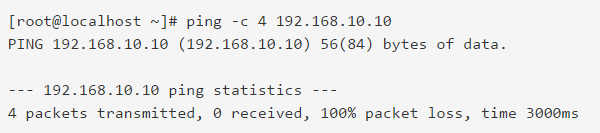
* ACCEPT:允许通过.
* LOG:记录日志信息,然后传给下一条规则继续匹配.
* REJECT:拒绝通过,必要时会给出提示.
* DROP:直接丢弃,不给出任何回应.

其中REJECT和DROP的操作都是将数据包拒绝，但REJECT会再回复一条“您的信息我已收到，但被扔掉了”。

通过ping命令测试REJECT情况会是这样的：



但如果是DROP则不予响应：

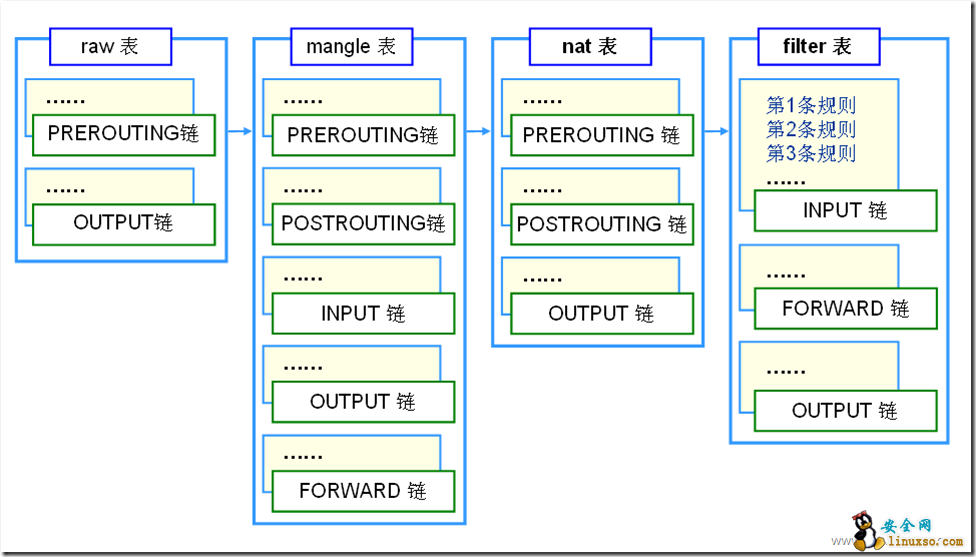


而规则链则依据处理数据包的位置不同而进行分类：

* PREROUTING: 在进行路由选择前处理数据包
* INPUT: 处理入站的数据包
* OUTPUT: 处理出站的数据包
* FORWARD: 处理转发的数据包
* POSTROUTING: 在进行路由选择后处理数据包

Iptables中的规则表是用于容纳规则链，规则表默认是允许状态的，那么规则链就是设置被禁止的规则，而反之如果规则表是禁止状态的，那么规则链就是设置被允许的规则。

* raw表:确定是否对该数据包进行状态跟踪
* mangle表:为数据包设置标记
* nat表:修改数据包中的源、目标IP地址或端口
* filter表:确定是否放行该数据包（过滤）



规则表的先后顺序:raw→mangle→nat→filter

规则链的先后顺序:

* 入站顺序:PREROUTING→INPUT
* 出站顺序:OUTPUT→POSTROUTING
* 转发顺序:PREROUTING→FORWARD→POSTROUTING

还有三点注意事项：

1.没有指定规则表则默认指filter表。

2.不指定规则链则指表内所有的规则链。

3.在规则链中匹配规则时会依次检查，匹配即停止（LOG规则例外），若没匹配项则按链的默认状态处理。

**2.2 基本的命令参数**

iptables命令用于管理防火墙的规则策略，格式为：

“**iptables [-t 表名] 选项 [链名] [条件] [-j 控制类型]**”。

表格为读者总结了几乎所有常用的iptables参数，如果记不住也没关系，用时来查就行，看完后来学习下如何组合并使用吧：

|  |  |
| --- | --- |
| **参数** | **作用** |
| -P | 设置默认策略:iptables -P INPUT (DROP|ACCEPT) |
| -F | 清空规则链 |
| -L | 查看规则链 |
| -A | 在规则链的末尾加入新规则 |
| -I num | 在规则链的头部加入新规则 |
| -D num | 删除某一条规则 |
| -s | 匹配来源地址IP/MASK，加叹号"!"表示除这个IP外。 |
| -d | 匹配目标地址 |
| -i 网卡名称 | 匹配从这块网卡流入的数据 |
| -o 网卡名称 | 匹配从这块网卡流出的数据 |
| -p | 匹配协议,如tcp,udp,icmp |
| --dport num | 匹配目标端口号 |
| --sport num | 匹配来源端口号 |

查看已有的规则：

[root@linuxprobe ~]# iptables -L

清空已有的规则：

[root@linuxprobe ~]# iptables -F

将INPUT链的默认策略设置为拒绝：

当INPUT链默认规则设置为拒绝时，我们需要写入允许的规则策略。

这个动作的目地是当接收到数据包时，按顺序匹配所有的允许规则策略，当全部规则都不匹配时，拒绝这个数据包。

[root@linuxprobe ~]# iptables -P INPUT DROP

允许所有的ping操作：

[root@linuxprobe ~]# iptables -I INPUT -p icmp -j ACCEPT

在INPUT链的末尾加入一条规则，允许所有未被其他规则匹配上的数据包：

因为默认规则表就是filter，所以其中的"-t filter"一般省略不写，效果是一样的。

[root@linuxprobe ~]# iptables -t filter -A INPUT -j ACCEPT

删除上面的那条规则：

[root@linuxprobe ~]# iptables -D INPUT 2

既然读者已经掌握了iptables命令的基本参数，那么来尝试解决模拟训练吧：

**模拟训练A:** 仅允许来自于192.168.10.0/24域的用户连接本机的ssh服务。

Iptables防火墙会按照顺序匹配规则，请一定要保证“允许”规则是在“拒绝”规则的上面。

[root@linuxprobe ~]# iptables -I INPUT -s 192.168.10.0/24 -p tcp --dport 22 -j ACCEPT

[root@linuxprobe ~]# iptables -A INPUT -p tcp --dport 22 -j REJECT

**模拟训练B:** 不允许任何用户访问本机的12345端口。

[root@linuxprobe ~]# iptables -I INPUT -p tcp --dport 12345 -j REJECT

[root@linuxprobe ~]# iptables -I INPUT -p udp --dport 12345 -j REJECT

**模拟实验C:** 拒绝其他用户从"eno16777736"网卡访问本机http服务的数据包。

[root@linuxprobe ~]# iptables -I INPUT -i en016777736 -p tcp --dport 80 -j REJECT

**模拟训练D:** 禁止用户访问www.my133t.org。

[root@linuxprobe ~]# iptables -I FORWARD -d www.my133t.org -j DROP

**模拟训练E:** 禁止IP地址是192.168.10.10的用户上网

[root@linuxprobe ~]# iptables -I FORWARD -s 192.168.10.10 -j DROP

ptables命令执行后的规则策略仅当前生效，若想重启后依然保存规则需执行

# service iptables save

另外流量均衡技术也是常用的技术，比如将一台主机作为网站的前端服务器，将访问流量分流至内网中3台不同的主机上。

iptables -A PREROUTING -i eth0 -p tcp --dport 80 -m state --state NEW -m nth --counter 0 --every 3 --packet 0 -j DNAT --to-destination 192.168.10.10:80

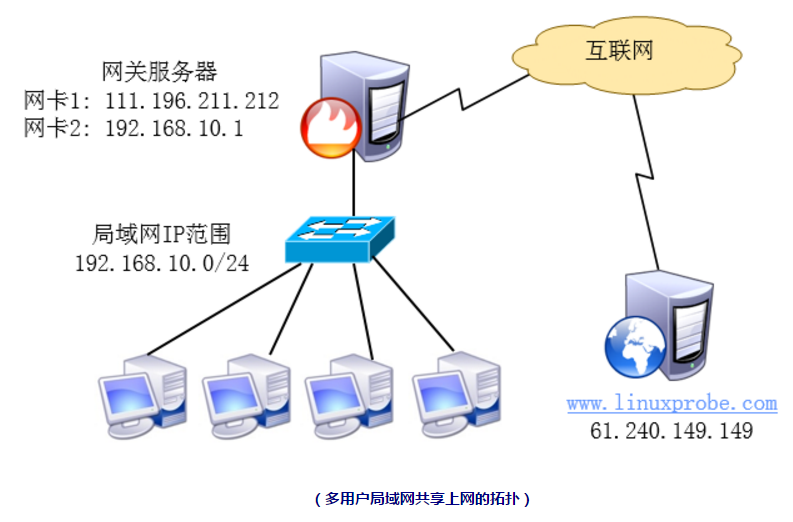
iptables -A PREROUTING -i eth0 -p tcp --dport 80 -m state --state NEW -m nth --counter 0 --every 3 --packet 0 -j DNAT --to-destination 192.168.10.11:80

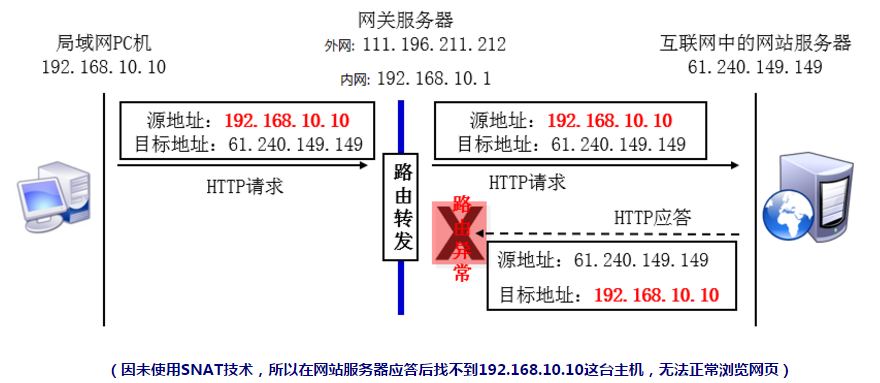
iptables -A PREROUTING -i eth0 -p tcp --dport 80 -m state --state NEW -m nth --counter 0 --every 3 --packet 0 -j DNAT --to-destination 192.168.10.12:80

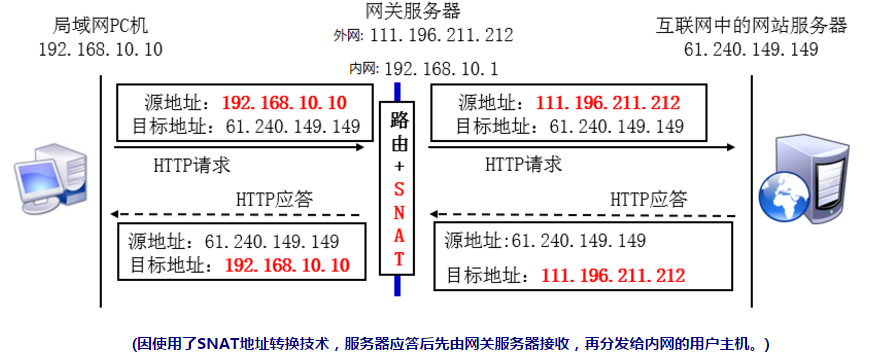
**2.3 SNAT与DNAT**

SNAT即源地址转换技术，能够让多个内网用户通过一个外网地址上网，解决了IP资源匮乏的问题，确实很实用。

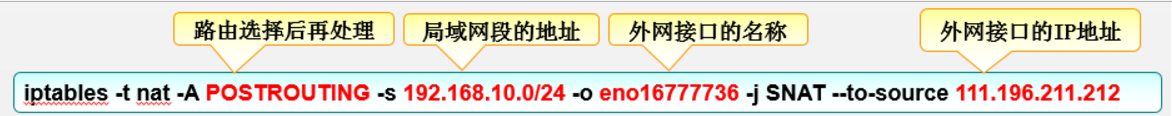
例如读者们来访问《Linux就该这么学》的网页，则就是通过家中的网关设备（您的无线路由器）进行的SNAT转换。







现在需要将"192.168.10.0"网段的内网IP用户经过地址转换技术变成外网IP地址"111.196.211.212"，这样一来内网IP用户就都可以通过这个外网IP上网了，使用iptables防火墙即可实现SNAT源地址转换，根据需求命令如下：

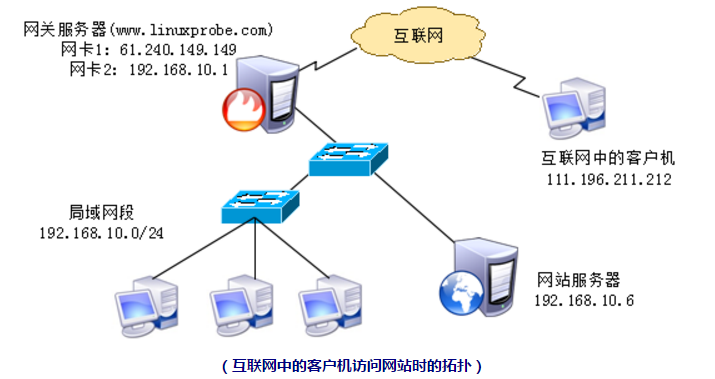


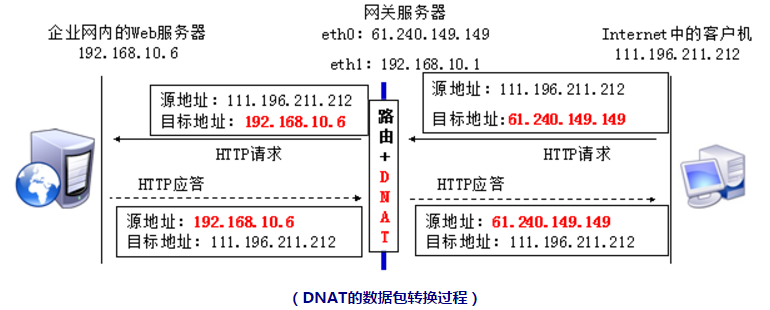
不知读者有无这种经历，当使用联通或者电信上网的时候，每次拨号都会重新分配新的IP地址，那么若网关IP经常变动怎么办？

这种外网IP地址不稳定的情况即可使用MASQUERADE(动态伪装):能够自动的寻找外网地址并改为当前正确的外网IP地址

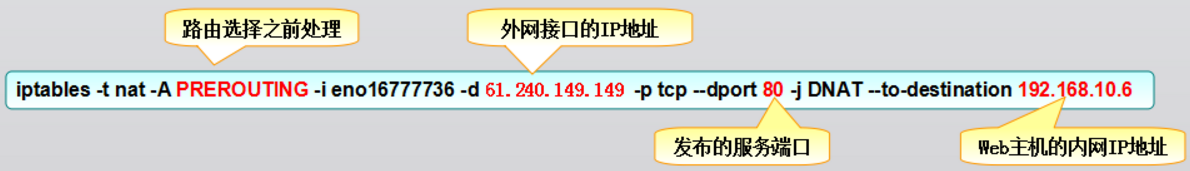
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -j MASQUERADE

**DNAT即目地地址转换技术**，则能够让外网IP用户访问局域网内不同的服务器。





现在希望互联网中的客户机访问到内网"192.168.10.6"这台提供网站服务的主机，那么只需在网关系统上运行这条命令：



**3 Firewalld防火墙**

Firewalld服务是红帽RHEL7系统中默认的防火墙管理工具，特点是拥有运行时配置与永久配置选项且能够支持动态更新以及"zone"的区域功能概念，使用图形化工具firewall-config或文本管理工具firewall-cmd，下面实验中会讲到~

**3.1 区域概念与作用**

防火墙的网络区域定义了网络连接的可信等级，我们可以根据不同场景来调用不同的firewalld区域，区域规则有：

|  |  |
| --- | --- |
| **区域** | **默认规则策略** |
| trusted | 允许所有的数据包。 |
| home | 拒绝流入的数据包，除非与输出流量数据包相关或是ssh,mdns,ipp-client,samba-client与dhcpv6-client服务则允许。 |
| internal | 等同于home区域 |
| work | 拒绝流入的数据包，除非与输出流量数据包相关或是ssh,ipp-client与dhcpv6-client服务则允许。 |
| public | 拒绝流入的数据包，除非与输出流量数据包相关或是ssh,dhcpv6-client服务则允许。 |
| external | 拒绝流入的数据包，除非与输出流量数据包相关或是ssh服务则允许。 |
| dmz | 拒绝流入的数据包，除非与输出流量数据包相关或是ssh服务则允许。 |
| block | 拒绝流入的数据包，除非与输出流量数据包相关。 |
| drop | 拒绝流入的数据包，除非与输出流量数据包相关。 |

简单来讲就是为用户预先准备了几套规则集合，我们可以根据场景的不同选择合适的规矩集合，而默认区域是public。

**3.2 字符管理工具**

如果想要更高效的配置妥当防火墙，那么就一定要学习字符管理工具firewall-cmd命令,命令参数有：

|  |  |
| --- | --- |
| 参数 | 作用 |
| --get-default-zone | 查询默认的区域名称。 |
| --set-default-zone=<区域名称> | 设置默认的区域，永久生效。 |
| --get-zones | 显示可用的区域。 |
| --get-services | 显示预先定义的服务。 |
| --get-active-zones | 显示当前正在使用的区域与网卡名称。 |
| --add-source= | 将来源于此IP或子网的流量导向指定的区域。 |
| --remove-source= | 不再将此IP或子网的流量导向某个指定区域。 |
| --add-interface=<网卡名称> | 将来自于该网卡的所有流量都导向某个指定区域。 |
| --change-interface=<网卡名称> | 将某个网卡与区域做关联。 |
| --list-all | 显示当前区域的网卡配置参数，资源，端口以及服务等信息。 |
| --list-all-zones | 显示所有区域的网卡配置参数，资源，端口以及服务等信息。 |
| --add-service=<服务名> | 设置默认区域允许该服务的流量。 |
| --add-port=<端口号/协议> | 允许默认区域允许该端口的流量。 |
| --remove-service=<服务名> | 设置默认区域不再允许该服务的流量。 |
| --remove-port=<端口号/协议> | 允许默认区域不再允许该端口的流量。 |
| --reload | 让“永久生效”的配置规则立即生效，覆盖当前的。 |

特别需要注意的是firewalld服务有两份规则策略配置记录，必需要能够区分：

* RunTime:当前正在生效的。
* Permanent:永久生效的。

当下面实验修改的是永久生效的策略记录时，必须执行"--reload"参数后才能立即生效，否则要重启后再生效。

查看当前的区域：

[root@linuxprobe ~]# firewall-cmd --get-default-zone

public

查询eno16777728网卡的区域：

[root@linuxprobe ~]# firewall-cmd --get-zone-of-interface=eno16777728

public

在public中分别查询ssh与http服务是否被允许：

[root@linuxprobe ~]# firewall-cmd --zone=public --query-service=ssh

yes

[root@linuxprobe ~]# firewall-cmd --zone=public --query-service=http

no

设置默认规则为dmz：

[root@linuxprobe ~]# firewall-cmd --set-default-zone=dmz

让“永久生效”的配置文件立即生效：

[root@linuxprobe ~]# firewall-cmd --reload

success

启动/关闭应急状况模式，阻断所有网络连接：

应急状况模式启动后会禁止所有的网络连接，一切服务的请求也都会被拒绝，当心，请慎用。

[root@linuxprobe ~]# firewall-cmd --panic-on

success

[root@linuxprobe ~]# firewall-cmd --panic-off

success

如果您已经能够完全理解上面练习中firewall-cmd命令的参数作用，不妨来尝试完成下面的模拟训练吧：

**模拟训练A:**允许https服务流量通过public区域，要求立即生效且永久有效：

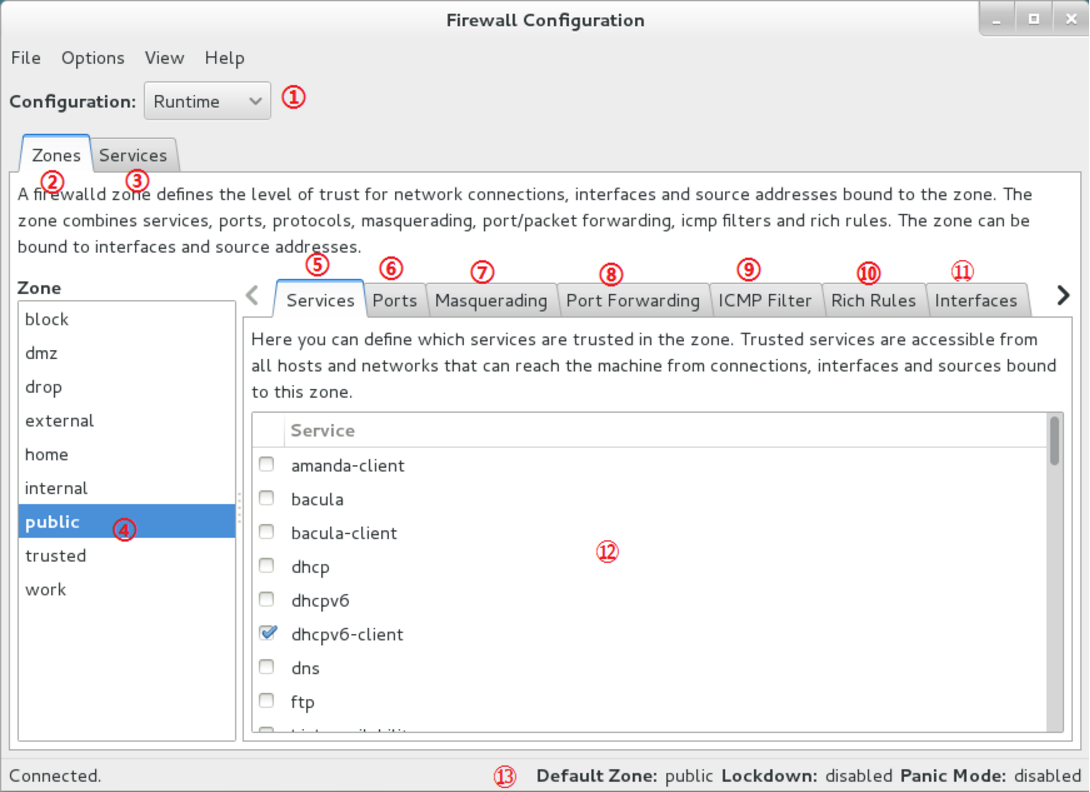
方法一:分别设置当前生效与永久有效的规则记录：

[root@linuxprobe ~]# firewall-cmd

**3.3 图形管理工具**

执行firewall-config命令即可看到firewalld的防火墙图形化管理工具，真的很强大，可以完成很多复杂的工作。

firewalld防火墙图形化管理工具界面详解：



①:选择"立即生效"或"重启后依然生效"配置。

②:区域列表。

③:服务列表。

④:当前选中的区域。

⑤:被选中区域的服务。

⑥:被选中区域的端口。

⑦:被选中区域的伪装。

⑧:被选中区域的端口转发。

⑨:被选中区域的ICMP包。

⑩:被选中区域的富规则。

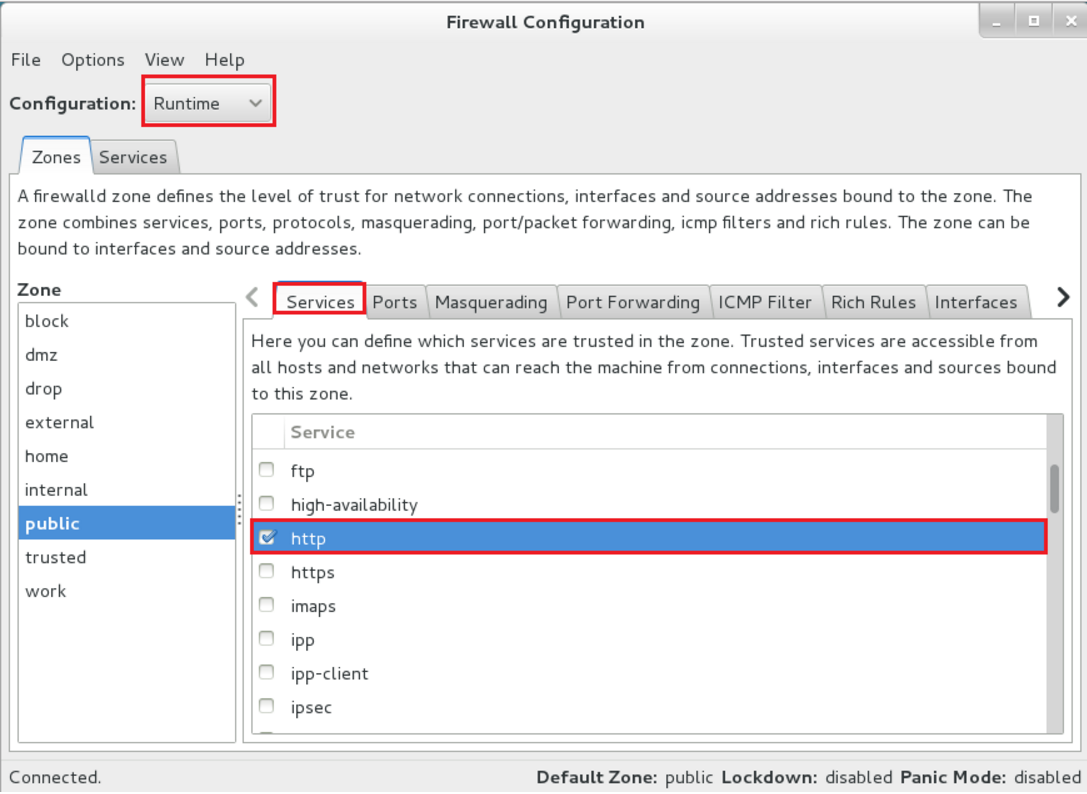
⑪:被选中区域的网卡设备。

⑫:被选中区域的服务，前面有√的表示允许。

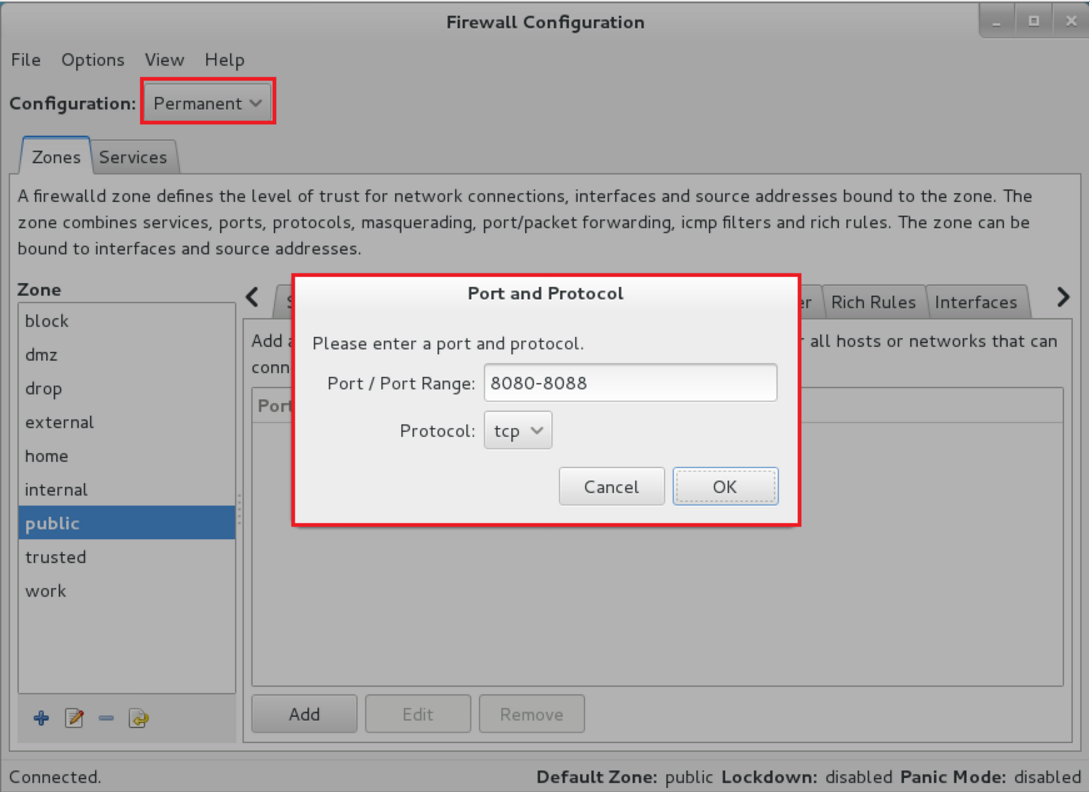
⑬:firewalld防火墙的状态

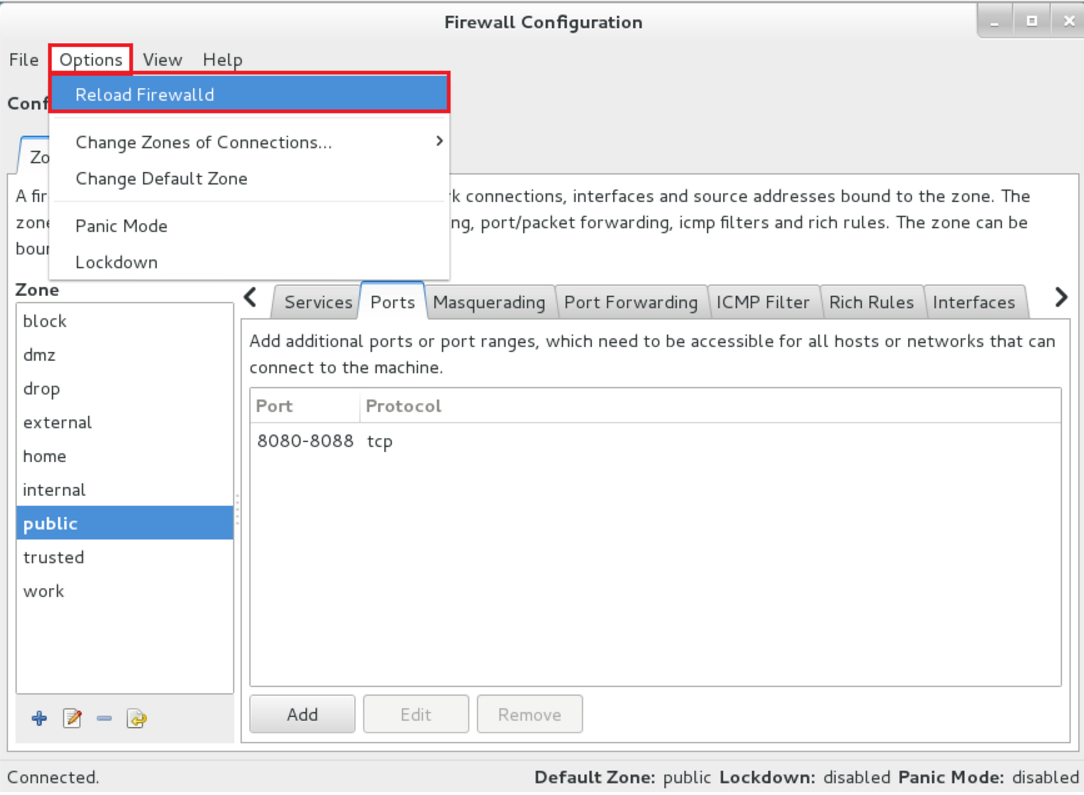
请注意:firewall-config图形化管理工具中没有保存/完成按钮，只要修改就会生效。

允许其他主机访问http服务，仅当前生效：



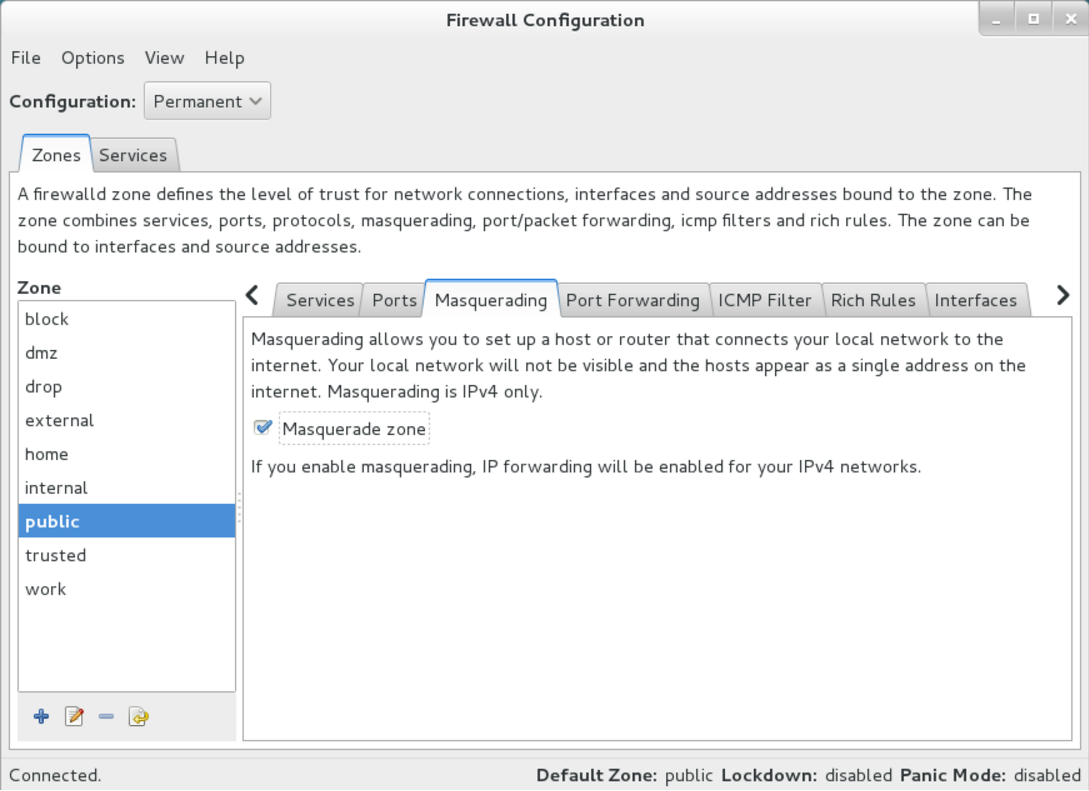
允许其他主机访问8080-8088端口且重启后依然生效：

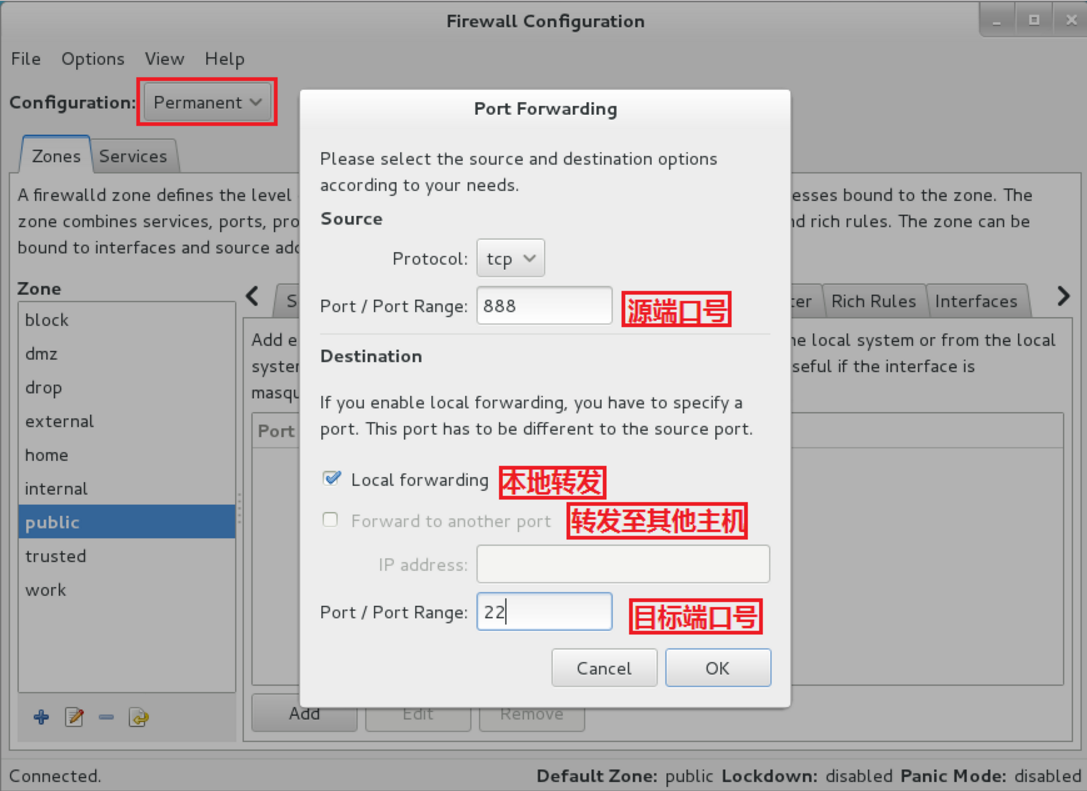




开启伪装功能，重启后依然生效：

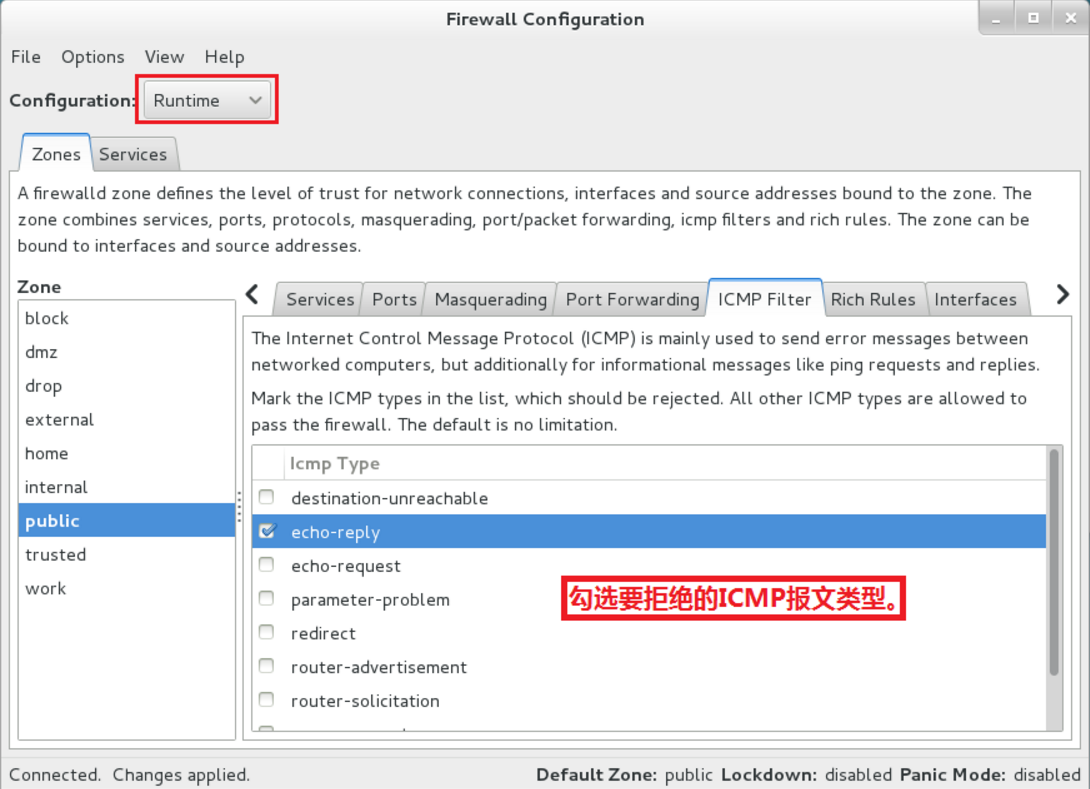
firewalld防火墙的伪装功能实际就是SNAT技术，即让内网用户不必在公网中暴露自己的真实IP地址。





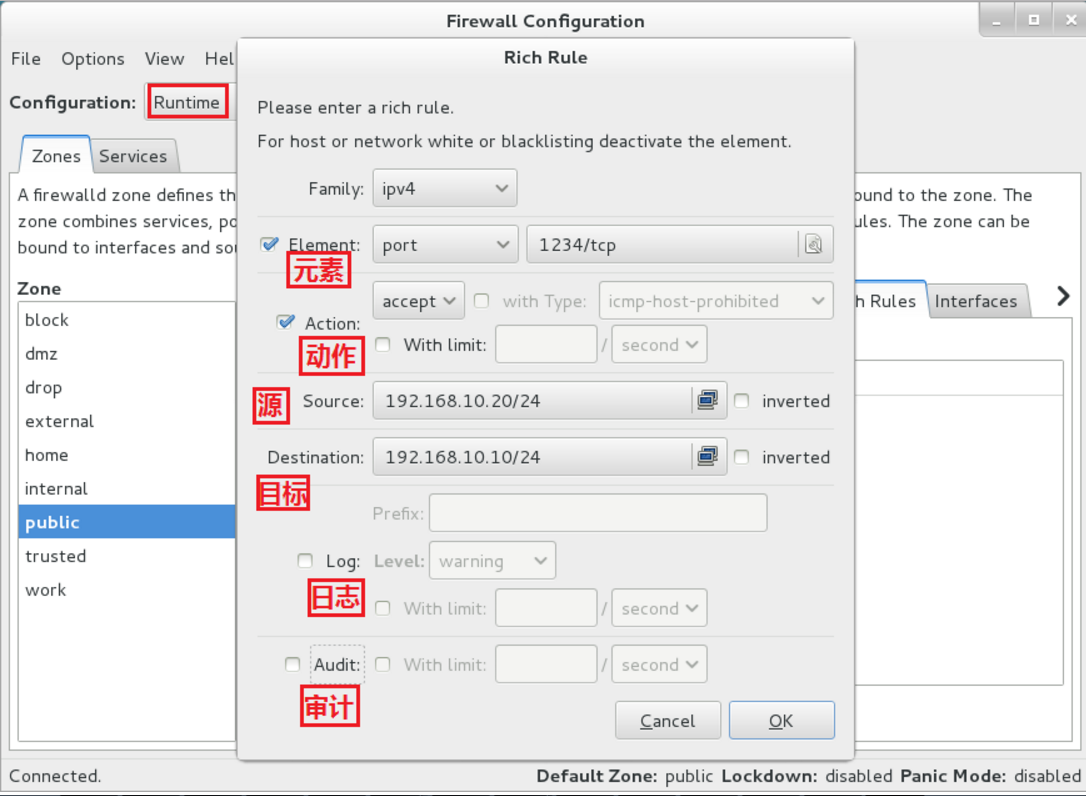
过滤所有"echo-reply"的ICMP协议报文数据包，仅当前生效：

ICMP即互联网控制报文协议"Internet Control Message Protocol"，归属于TCP/IP协议族，主要用于检测网络间是否可通信、主机是否可达、路由是否可用等网络状态，并不用于传输用户数据。

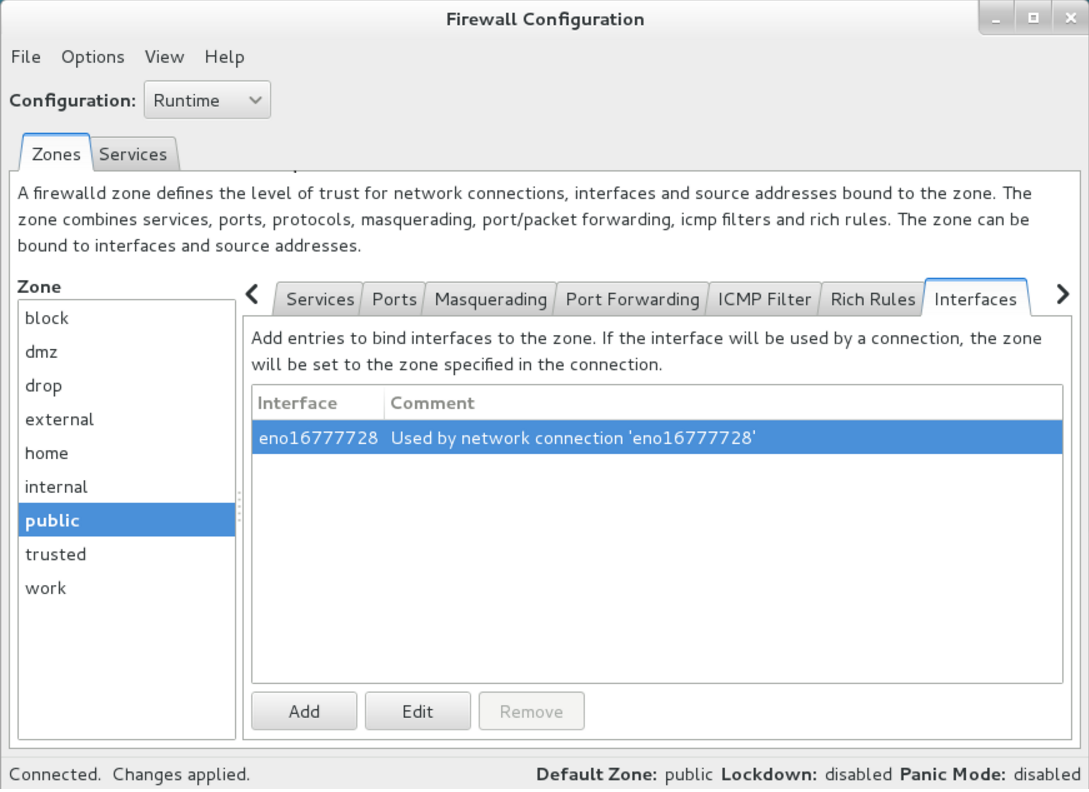


仅允许192.168.10.20主机访问本机的1234端口，仅当前生效：

富规则代表着更细致、更详细的规则策略，针对某个服务、主机地址、端口号等选项的规则策略，优先级最高。



查看网卡设备信息：



firewall-config图形管理工具真的非常实用，很多原本复杂的长命令被用图形化按钮替代，设置规则也变得简单了，日常工作中真的非常实用。所以有必要跟读者们讲清配置防火墙的原则——只要能实现需求的功能，无论用文本管理工具还是图形管理工具都是可以的。

**4 服务的访问控制列表**

Tcp\_wrappers(即Transmission Control Protocol(TCP)Wrappers)是一款基于IP层的ACL访问控制列表流量监控程序，它能够根据来访主机地址与本机目标服务程序做允许或拒绝规则，控制列表修改后会立即生效，系统将会先检查允许规则，如果匹配允许则直接放行流量，若拒绝规则中匹配则直接拒绝，都不匹配默认也会放行。

* 允许名单:/etc/hosts.allow
* 拒绝名单:/etc/hosts.deny

指定客户端的规则如下：

|  |  |  |
| --- | --- | --- |
| **客户端类型** | **示例** | **满足示例的客户端列表** |
| 单一主机 | 192.168.10.10 | IP地址为192.168.10.10的主机。 |
| 指定网段 | 192.168.10. | IP段为192.168.10.0/24的主机。 |
| 指定网段 | 192.168.10.0/255.255.255.0 | IP段为192.168.10.0/24的主机。 |
| 指定DNS后缀 | .linuxprobe.com | 所有DNS后缀为.linuxprobe.com的主机 |
| 指定主机名称 | boss.linuxprobe.com | 主机名称为boss.linuxprobe.com的主机。 |
| 指定所有客户端 | ALL | 所有主机全部包括在内。 |

限制只有192.168.10.0/24网段的主机可以访问本机的sshd服务：

编辑允许规则：

[root@linuxprobe ~]# vim /etc/hosts.allow

sshd:192.168.10.

拒绝其他所有的主机：

[root@linuxprobe ~]# vim /etc/hosts.deny

sshd:\*