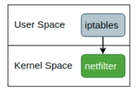
Linux iptables

基本概念

iptables与netfilter

如果Google一下"Linux如何关闭防火墙"，答案都是清一色的 `service iptables stop`，那么iptables是否就是Linux的防火墙呢？

实际上，iptables仅仅只是位于用户空间的一个命令行工具，真正的防火墙框架是位于内核中的 netfilter，就如同redis-cli和redis-server之间的关系一样



netfilter

- netfilter为Linux内置的、免费的防火墙模块
- 作用
  - 数据包过滤
  - 网络地址转换(NAT)
  - 数据包内容修改

iptables规则

防火墙的基本原理其实就是规则的过滤，当某一个数据包符合或者不符合相关的规则时，防火墙可以丢弃、拒绝或者放行该数据包。而规则的制定可以根据网络包的源IP地址、目的IP地址、源端口号、目的端口号、传输类型(TCP/UDP)和服务类型(HTTP、FTP)来多方位的定制

以接收TCP数据包为例，我们为接收数据包这一场景添加多个规则，例如源IP地址为10.23.36.98的数据包直接丢弃，目的端口号为7070的数据包也丢弃，这些规则将会组成一个规则列表

一台主机网络会有相当多的场景，例如上面的接收数据包，同时还会有发送数据包，转发数据包，这些场景在iptables中称之为"链"

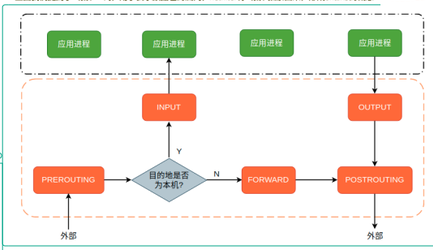
为什么是链？ 多个规则组合起来，就像一个链条一样，每个数据包都要匹配链上的所有规则，才能最终决定该数据包的结果

规则 = 匹配条件 + 动作

- 匹配条件：源/目的IP地址，源/目的端口号
- 动作
  - ACCEPT：允许数据包通过
  - DROP：直接丢弃该数据包
  - REJECT：拒绝数据包通过，必要时会给数据发送端一个响应的信息
  - SNAT：源IP地址转换，多用于内网用户使用同一个公网IP地址的情况
  - DNAT：目的IP地址转换
  - MASQUERADE：是SNAT的一种特殊形式，适用于动态的、临时会变的IP上

netfilter chains

上面我们提到了"场景"一词，用于表示数据包的流向，netfilter将"场景"抽象出来，形成chains的概念



- ROUTING：接收的数据包刚进入内核，未经过路由校验，即不知道到底是发送给本机的，还是其它机器
- INPUT：经过了路由选择，并且目的地为本机，则进入本地数据包处理流程
- FORWARD：经过了路由选择，并且目的地为其它机器，进入转发流程
- OUTPUT：本地应用进程将要发出的数据包
- POSTROUTING：本地应用进程发出的数据包，或者转发的数据包所需要经过的路由选择

通过上图我们可以很直观的得到一些路径

- HTTP请求：PREROUTING -> INPUT
- HTTP转发请求：PREROUTING -> FORWARD -> POSTROUTING
- HTTP响应：OUTPUT -> POSTROUTING

表

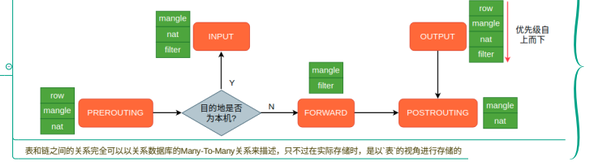
在每一个链上面都可以制定多个规则(rule)，netfilter将规则具体化、结构化，使得用户只能定义一些特定的规则

- filter：用于定义数据包的过滤规则
- nat：用于定义网络地址转换(NAT)
- mangle：用于定义对报文的修改
- raw：关闭nat表上启用的连接追踪机制

并不是所有的链都能制定这四种规则

表链关系

正如前面所提到了，链不一定能够定义上述的全部四种规则，事实上，除了OUTPUT链能够定义全部规则以外，其余链均只能定义三种或两种规则



在实际操作iptables时，实际操作的是表，而不是链

表和链之间的关系完全可以以关系数据库的Many-To-Many关系来描述，只不过在实际存储时，是以'表'的视角进行存储的

iptables CURD(增删改查)

操作系统本身就是CURD，更何况是iptables呢？只要你眼中有CRUD，万物皆可CURD _(:3」∠)_

查询

前面提到了表和链是一个多对多的关系，所以在查询规则时会将表和链一起制定

- -t (table)：指定表
- -L (List)：指定链

例如，查询POSTROUTING链上关于nat相关的规则：sudo iptables -t nat -L POSTROUTING

```
smartkeyerror@Zero:~$ sudo iptables -t nat -vL POSTROUTING
Chain POSTROUTING (policy ACCEPT 8584K packets, 577M bytes)
 pkts bytes target     prot opt in     out     source               destination
   89  5856 MASQUERADE  all  --  any    !docker0  172.17.0.0/16        anywhere
    0     0 MASQUERADE  all  --  any    !br-9ffb472a4b61  172.18.0.0/16        anywhere
```

- pkts (packages)：当前规则匹配到的报文总数
- bytes：当前规则匹配到的报文大小总和
- target：规则匹配后所做的动作
- prot (protocol)：协议类型
- opt (option)：规则对应的相关选项
- int：数据包由哪个接口(网卡)流入，我们可以设置通过哪块网卡流入的报文需要匹配当前规则
- out：数据包由哪个接口(网卡)流出，我们可以设置通过哪块网卡流出的报文需要匹配当前规则
- source：规则对应的源头地址，可以是一个IP，也可以是一个网段
- destination：规则对应的目标地址，可以是一个IP，也可以是一个网段

新增

sudo iptables -t filter -I INPUT -s 192.168.2.3 -j DROP：该条命令表示，在filter表中添加一条关于INPUT链的规则，规则内容是来源IP为192.168.2.3的所有数据包均丢弃

新增规则主要用到这些命令

- -I：即Insert，在链的头部插入规则
- -A：即Append，在链的尾部添加规则
- -s：来源IP地址或者网段
- -d：目标IP地址或者网段
- -j：即 jump-to，定义匹配规则后的动作

另外一个需要注意的是，iptables的匹配规则是自上而下的

删除

根据规则编号进行删除：sudo iptables -t filter -D INPUT 2：删除filter表中INPUT链的编号为2的规则

根据具体的匹配条件与动作删除：sudo iptables -t filter -D INPUT -s 192.168.2.3 -j DROP：删除filter表中INPUT链中源IP地址为192.168.2.3，且匹配动作为DROP的规则

修改

在不完全熟悉iptables修改规则的前提下，禁止使用 `-R` 命令进行修改。而应该采用先删除、后新增的方式

在使用 `-R` 参数修改规则时，必须指定原有规则的匹配条件