Linux 防火墙 iptables 学习笔记（三）

```
iptables-F

iptables-X

iptables-F-tmangle

iptables-tmangle-X

iptables-F-tnat

iptables-tnat-X
```

首先，把三个表清空，把自建的规则清空。

```
iptables-PINPUTDROP

iptables-POUTPUTDROP

iptables-PFORWARDACCEPT
```

设定 INPUT、OUTPUT 的默认策略为 DROP，FORWARD 为 ACCEPT。

```
iptables-AINPUT-ilo-jACCEPT

iptables-AOUTPUT-olo-jACCEPT
```

先把“回环”打开，以免有不必要的麻烦。

```
    iptables-AINPUT-ieth+-picmp--icmp-type8-jACCEPT

    iptables-AOUTPUT-oeth+-picmp--icmp-type0-jACCEPT
```

在所有网卡上打开 ping 功能，便于维护和检测。

```
    iptables-AINPUT-ieth0-s192.168.100.250-d192.168.100.1-
ptcp--dport22-jACCEPT

    iptables-AOUTPUT-oeth0-d192.168.100.250-s192.168.100.1
-ptcp--sport22-jACCEPT
```

打开 22 端口，允许远程管理。（设定了很多的附加条件：管理机器 IP 必须是 250，并且必须从 eth0 网卡进入）

```
    iptables-AINPUT-ieth0-s192.168.100.0/24-ptcp--dport312
8-mstate--stateNEW,ESTABLISHED-jACCEPT

    iptables-AOUTPUT-oeth0-d192.168.100.0/24-ptcp--sport31
28-mstate--stateESTABLISHED-jACCEPT

    iptables-AINPUT-ieth1-s192.168.168.0/24-ptcp--dport312
8-mstate--stateNEW,ESTABLISHED-jACCEPT

    iptables-AOUTPUT-oeth1-d192.168.168.0/24-ptcp--sport31
28-mstate--stateESTABLISHED-jACCEPT

    iptables-AINPUT-ieth2-ptcp--dport32768:61000-mstate--s
tateESTABLISHED-jACCEPT

    iptables-AOUTPUT-oeth2-ptcp--sport32768:61000-mstate--
stateNEW,ESTABLISHED-jACCEPT
```

```
    iptables-AOUTPUT-oeth2-pudp--dport53-jACCEPT

    iptables-AINPUT-ieth2-pudp--sport53-jACCEPT
```

上面这几句是比较头痛的，我做逐一解释。

```
    iptables-AINPUT-ieth0-s192.168.100.0/24-ptcp--dport312
8-mstate--stateNEW,ESTABLISHED-jACCEPT
```

允许 192.168.100.0/24 网段的机器发送数据包从 eth0 网卡进入。如果数据包是 tcp 协议，而且目的端口是 3128（因为 REDIRECT 已经把 80 改为 3128 了。nat 表的 PREROUTING 是在 filter 表的 INPUT 前面的。）的，再而且，数据包的状态必须是 NEW 或者 ESTABLISHED 的（NEW 代表 tcp 三段式握手的“第一握”，换句话说就是，允许客户端机器向服务器发出链接申请。ESTABLISHED 表示通过握手已经建立起链接），通过。

```
    iptables-AOUTPUT-oeth2-ptcp--sport32768:61000-mstate--
stateNEW,ESTABLISHED-jACCEPT
```

我们先来看这一句。现在你的数据包已经进入到 linux 服务器防火墙上来了。squid 需要代替你去访问，所以这时，服务器就成了客户端的角色，所以它要使用 32768 到 61000 的私有端口进行访问。（大家会奇怪应该是 1024 到 65535 吧。其实 CentOS 版的 linux 所定义的私有端口是 32768 到 61000 的，你可以通过 cat/proc/sys/net/ipv4/ip_local_port_range，查看一下。）再次声明：这里是

squid 以客户端的身份去访问其他的服务器 ,所以这里的源端口是 32768:61000，而不是 3128！

```
    iptables-AINPUT-ieth2-ptcp--dport32768:61000-mstate--stateESTABLISHED-jACCEPT
```

当然了，数据有去就有回。

```
    iptables-AOUTPUT-oeth0-d192.168.100.0/24-ptcp--sport3128-mstate--stateESTABLISHED-jACCEPT
```

数据包还得通过服务器，转到内网网卡上。请注意，这里，是 squid 帮你去访问了你想要访问的网站。所以在内网中，你的机器是客户端角色，而 squid 是服务器角色。这与刚才对外访问的过程是不同的。所以在这里，源端口是 3128，而不是 32768:61000。

```
    iptables-AOUTPUT-oeth2-pudp--dport53-jACCEPT

    iptables-AINPUT-ieth2-pudp--sport53-jACCEPT
```

当然，DNS 是不可缺少的。

```
    iptables-AINPUT-ieth+-ptcp--dport80-jLOG--log-prefix"iptables_80_alert"--log-levelinfo

    iptables-AINPUT-ieth+-ptcp--dport21-jLOG--log-prefix"iptables_21_alert"--log-levelinfo
```

```
    iptables-AINPUT-ieth+-ptcp--dport22-jLOG--log-prefix"iptables_22_alert"--log-levelinfo

    iptables-AINPUT-ieth+-ptcp--dport25-jLOG--log-prefix"iptables_25_alert"--log-levelinfo

    iptables-AINPUT-ieth+-picmp--icmp-type8-jLOG--log-prefix"iptables_icmp8_alert"--log-levelinfo
```

当然了，来点日志记录会对网管员有所帮助。

iptables 基本命令使用举例

一、链的基本操作

1、清除所有的规则。

1）清除预设表 filter 中所有规则链中的规则。

```
    #iptables-F
```

2）清除预设表 filter 中使用者自定链中的规则。

```
    #iptables-X

    #iptables-Z
```

2、设置链的默认策略。一般有两种方法。

1）首先允许所有的包，然后再禁止有危险的包通过放火墙。

```
#iptables-PINPUTACCEPT

#iptables-POUTPUTACCEPT

#iptables-PFORWARDACCEPT
```

2）首先禁止所有的包，然后根据需要的服务允许特定的包通过防火墙。

```
#iptables-PINPUTDROP

#iptables-POUTPUTDROP

#iptables-PFORWARDDROP
```

3、列出表/链中的所有规则。默认只列出 filter 表。

```
#iptables-L
```

4、向链中添加规则。下面的语句用于开放网络接口：

```
#iptables-AINPUT-ilo-jACCEPT

#iptables-AOUTPUT-olo-jACCEPT

#iptables-AINPUT-ieth0-jACEPT

#iptables-AOUTPUT-oeth1-jACCEPT

#iptables-AFORWARD-ieth1-jACCEPT

#iptables-AFORWARD-0eth1-jACCEPT
```

注意:由于本地进程不会经过 FORWARD 链，因此回环接口 lo 只在 INPUT 和 OUTPUT 两个链上作用。

5、使用者自定义链。

```
#iptables-Ncustom

#iptables-Acustom-s0/0-d0/0-picmp-jDROP

#iptables-AINPUT-s0/0-d0/0-jDROP
```

二、设置基本的规则匹配

1、指定协议匹配。

1）匹配指定协议。

```
#iptables-AINPUT-ptcp
```

2）匹配指定协议之外的所有协议。

```
#iptables-AINPUT-p!tcp
```

2、指定地址匹配。

1）指定匹配的主机。

```
#iptables-AINPUT-s192.168.0.18
```

2）指定匹配的网络。

```
#iptables-AINPUT-s192.168.2.0/24
```

3）匹配指定主机之外的地址。

```
#iptables-AFORWARD-s!192.168.0.19
```

4）匹配指定网络之外的网络。

```
#iptables-AFORWARD-s!192.168.3.0/24
```

3、指定网络接口匹配。

1）指定单一的网络接口匹配。

```
#iptables-AINPUT-ieth0

#iptables-AFORWARD-oeth0
```

2）指定同类型的网络接口匹配。

```
#iptables-AFORWARD-oppp+
```

4、指定端口匹配。

1）指定单一端口匹配。

```
#iptables-AINPUT-ptcp--sportwww

#iptables-AINPUT-pudp-dport53
```

2）匹配指定端口之外的端口。

```
#iptables-AINPUT-ptcp-dport!22
```

3）匹配端口范围。

```
#iptables-AINPUT-ptcp-sport22:80
```

4）匹配 ICMP 端口和 ICMP 类型。

```
#iptables-AINOUT-picmp-icimp-type8
```

5）指定 ip 碎片。

每个网络接口都有一个 MTU（最大传输单元），这个参数定义了可以通过的数据包的最大尺寸。如果一个数据包大于这个参数值时，系统会将其划分成更小的数据包

（称为 ip 碎片）来传输，而接受方则对这些 ip 碎片再进行重组以还原整个包。这样会导致一个问题：当系统将大数据包划分成 ip 碎片传输时，第一个碎片含有

完整的包头信息（IP+TCP、UDP 和 ICMP），但是后续的碎片只有包头的部分信息（如源地址、目的地址）。因此，检查后面的 ip 碎片的头部（象有

TCP、UDP 和 ICMP 一样）是不可能的。假如有这样的一条规则：

```
    #iptables-AFORWARD-ptcp-s192.168.1.0/24-d192.168.2.100
-dport80-jACCEPT
```

并且这时的 FORWARD 的 policy 为 DROP 时，系统只会让第一个 ip 碎片通过，而余下的碎片因为包头信息不完整而无法通过。可以通过—fragment/-f 选项来指定第二个及以后的 ip 碎片解决上述问题。

```
    #iptables-AFORWARD-f-s192.168.1.0/24-d192.168.2.100-jA
CCEPT
```

注意现在有许多进行 ip 碎片攻击的实例，如 DoS 攻击，因此允许 ip 碎片通过是有安全隐患的，对于这一点可以采用 iptables 的匹配扩展来进行限制。

三、设置扩展的规则匹配（举例已忽略目标动作）

1、多端口匹配。

1）匹配多个源端口。

```
    #iptables-AINPUT-ptcp-mmultiport-sport22,53,80,110
```

2）匹配多个目的端口。

```
#iptables-AINPUT-ptcp-mmultiport-dpoort22,53,80
```

3）匹配多端口(无论是源端口还是目的端口）

```
#iptables-AINPUT-ptcp-mmultiport-port22,53,80,110
```

2、指定 TCP 匹配扩展

使用–tcp-flags 选项可以根据 tcp 包的标志位进行过滤。

```
#iptables-AINPUT-ptcp-tcp-flagsSYN,FIN,ACKSYN

#iptables-AFROWARD-ptcp-tcp-flagsALLSYN,ACK
```

上实例中第一个表示 SYN、ACK、FIN 的标志都检查，但是只有 SYN 匹配。第二个表示 ALL（SYN，ACK，FIN，RST，URG，PSH）的标志都检查，但是只有设置了 SYN 和 ACK 的匹配。

```
#iptables-AFORWARD-ptcp--syn
```

选项—syn 相当于”--tcp-flagsSYN,RST,ACKSYN”的简写。

3、limit 速率匹配扩展。

1）指定单位时间内允许通过的数据包个数，单位时间可以是/second、/minute、/hour、/day 或使用第一个子母。

```
#iptables-AINPUT-mlimit--limit300/hour
```

```
2)指定触发事件的阀值。

#iptables-AINPUT-mlimit-limit-burst10
```

用来比对一次同时涌入的封包是否超过 10 个 超过此上限的包将直接丢弃。

3）同时指定速率限制和触发阀值。

```
#iptables-AINPUT-picmp-mlimit--limit3/m-limit-burst3
```

表示每分钟允许的最大包数量为限制速率（本例为 3）加上当前的触发阀值 burst 数。任何情况下，都可保证 3 个数据包通过，触发阀值 burst 相当于允许额外的包数量。

4）基于状态的匹配扩展（连接跟踪）

每个网络连接包括以下信息：源地址、目标地址、源端口、目的端口，称为套接字对（socketpairs）；协议类型、连接状态（TCP 协议）

和超时时间等。防火墙把这些信息称为状态（stateful）。状态包过滤防火墙能在内存中维护一个跟踪状态的表，比简单包过滤防火墙具有更大的安全性，命令格式如下：

```
iptables-mstate--state[!]state[,state,state,state]
```

其中，state 表是一个逗号分割的列表，用来指定连接状态，4 种：

>NEW:该包想要开始一个新的连接（重新连接或连接重定向）

>RELATED:该包是属于某个已经建立的连接所建立的新连接。举例：

FTP 的数据传输连接和控制连接之间就是 RELATED 关系。

>ESTABLISHED：该包属于某个已经建立的连接。

>INVALID:该包不匹配于任何连接，通常这些包被 DROP。

例如：

（1）在 INPUT 链添加一条规则，匹配已经建立的连接或由已经建立的连接所建立的新连接。即匹配所有的 TCP 回应包。

```
#iptables-AINPUT-mstate-stateRELATED,ESTABLISHED
```

（2）在 INPUT 链链添加一条规则，匹配所有从非 eth0 接口来的连接请求包。

```
#iptables-AINPUT-mstate--stateNEW-i!eth0
```

又如，对于 ftp 连接可以使用下面的连接跟踪：

（1）被动（Passive）ftp 连接模式。

```
#iptables-AINPUT-ptcp--sport1024:--dport1024:-mstate--stateESTABLISHED-jACCEPT
```

```
    #iptables-AOUTPUT-ptcp--sport1024:--dport1024:-m

    state--stateESTABLISHED,RELATED-jACCEPT
```

（2）主动（Active）ftp 连接模式

```
    #iptables-AINNPUT-ptcp--sport20-mstate--stateESTABLISH
ED,RELATED-jACCEPT

    #iptables-AOUTPUT-ptcp-OUTPUT-ptcp-dport20-mstate--sta
teESTABLISHED-jACCEPT
```