

iptables详解（5）：iptables匹配条件总结之二（常用扩展模块）

在本博客中，从理论到实践，系统的介绍了iptables，如果你想要从头开始了解iptables，可以查看iptables文章列表，直达链接如下

[iptables零基础快速入门系列](#)

前文已经总结了iptables中的基本匹配条件，以及简单的扩展匹配条件，此处，我们来认识一些新的扩展模块。

iprange扩展模块

之前我们已经总结过，在不使用任何扩展模块的情况下，使用-s选项或者-d选项即可匹配报文的源地址与目标地址，而且在指定IP地址时，可以同时指定多个IP地址号"隔开，但是，-s选项与-d选项并不能一次性的指定一段连续的IP地址范围，如果我们指定一段连续的IP地址范围，可以使用iprange扩展模块。

使用iprange扩展模块可以指定"一段连续的IP地址范围"，用于匹配报文的源地址或者目标地址。

iprange扩展模块中有两个扩展匹配条件可以使用

--src-range

--dst-range

没错，见名知意，上述两个选项分别用于匹配报文的源地址所在范围与目标地址所在范围。

示例如下：

```
[www.zsythink.net]#iptables -t filter -F INPUT
[www.zsythink.net]#iptables -t filter -I INPUT -m iprange --src-range 192.168.1.127-192.168.1.146 -j DROP
[www.zsythink.net]#iptables -nL INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
DROP       all  --  0.0.0.0/0              0.0.0.0/0              source IP range 192.168.1.127-192.168.1.146
[www.zsythink.net]#
```

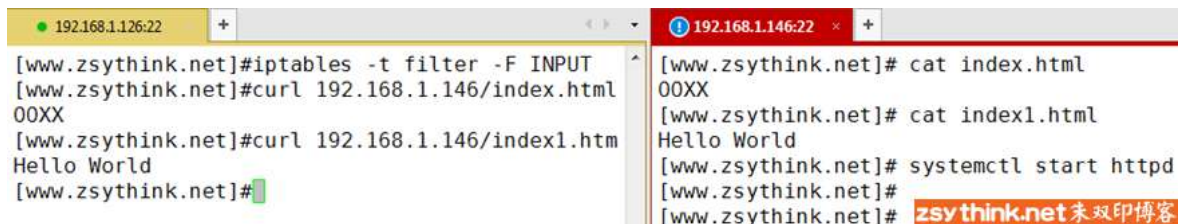
上例表示如果报文的源IP地址如果在192.168.1.127到192.168.1.146之间，则丢弃报文，IP段的始末IP使用"横杠"连接，--src-range与--dst-range和其他匹配条件用"!"取反，有了前文中的知识作为基础，此处就不再赘述了。

string扩展模块

使用string扩展模块，可以指定要匹配的字符串，如果报文中包含对应的字符串，则符合匹配条件。

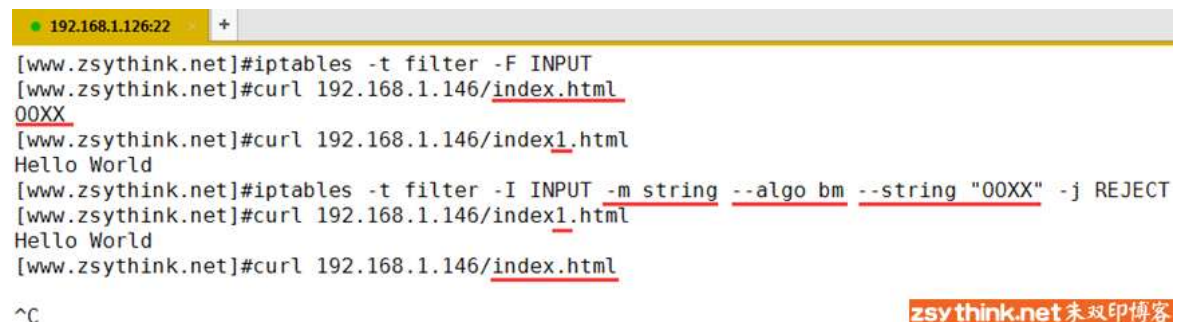
比如，如果报文中包含字符"OOXX"，我们就丢弃当前报文。

首先，我们在IP为146的主机上启动http服务，然后在默认的面目录中添加两个页面，页面中的内容分别为"OOXX"和"Hello World"，如下图所示，在没有配置126主机可以正常访问146主机上的这两个页面。



The screenshot shows two terminal windows. The left window is on host 192.168.1.126:22 and shows the execution of iptables rules to block traffic to 192.168.1.146/index.html and the successful curl of 192.168.1.146/index1.html. The right window is on host 192.168.1.146:22 and shows the contents of index.html (OOXX) and index1.html (Hello World), and the successful start of the httpd service.

那么，我们想要达到目的是，如果报文中包含"OOXX"字符，我们就拒绝报文进入本机，所以，我们可以在126上进行如下配置。



The screenshot shows a terminal window on host 192.168.1.126:22. It shows the iptables rule being added: iptables -t filter -I INPUT -m string --algo bm --string "OOXX" -j REJECT. It also shows the curl command for 192.168.1.146/index.html being executed, which would now be blocked.

上图中，'-m string'表示使用string模块，'--algo bm'表示使用bm算法去匹配指定的字符串，'--string "OOXX"'则表示我们想要匹配的字符串为"OOXX"。设置完上图中的规则后，由于index.html中包含"OOXX"字符串，所以，146的回应报文无法通过126的INPUT链，所以无法获取到页面对应的内容。

那么，我们来总结一下string模块的常用选项

--algo：用于指定匹配算法，可选的算法有bm与kmp，此选项为必须选项，我们不用纠结于选择哪个算法，但是我们必须指定一个。

--string：用于指定需要匹配的字符串。

time扩展模块

我们可以通过time扩展模块，根据时间段匹配报文，如果报文到达的时间在指定的时间范围以内，则符合匹配条件。

比如，“我想要自我约束，每天早上9点到下午6点不能看网页”，擦，多么残忍的规定，如果你想要这样定义，可以尝试使用如下规则。

```
#iptables -t filter -I OUTPUT -p tcp --dport 80 -m time --timestart 09:00:00 --timestop 18:00:00 -j REJECT
#iptables -t filter -I OUTPUT -p tcp --dport 443 -m time --timestart 09:00:00 --timestop 18:00:00 -j REJECT
#
```

上图中“-m time”表示使用time扩展模块，--timestart选项用于指定起始时间，--timestop选项用于指定结束时间。

如果你想要换一种约束方法，只有周六日不能看网页，那么可以使用如下规则。

```
#iptables -t filter -I OUTPUT -p tcp --dport 80 -m time --weekdays 6,7 -j REJECT
#
#
```

没错，如你所见，使用--weekdays选项可以指定每个星期的具体哪一天，可以同时指定多个，用逗号隔开，除了能够数字表示“星期几”，还能用缩写表示，例如：M Thu, Fri, Sat, Sun

当然，你也可以将上述几个选项结合起来使用，比如指定只有周六日的早上9点到下午6点不能浏览网页。

```
#iptables -t filter -I OUTPUT -p tcp --dport 80 -m time --timestart 09:00:00 --timestop 18:00:00 --weekdays 6,7 -j REJECT
#
#
```

聪明如你一定想到了，既然有--weekdays选项了，那么有没有--monthdays选项呢？必须有啊！

使用--monthdays选项可以具体指定的每个月的哪一天，比如，如下图设置表示指明每月的22号，23号。

```
#iptables -t filter -I OUTPUT -p tcp --dport 80 -m time --monthdays 22,23 -j REJECT
#
```

前文已经总结过，当一条规则中同时存在多个条件时，多个条件之间默认存在“与”的关系，所以，下图中的设置表示匹配的时间必须为星期五，并且这个“星期五”同时是每月的22号到28号之间的一天，所以，下图中的设置表示每个月的第4个星期五

```
#iptables -t filter -I OUTPUT -p tcp --dport 80 -m time --weekdays 5 --monthdays 22,23,24,25,26,27,28 -j REJECT
#
#
```

除了使用--weekdays选项与--monthdays选项，还可以使用--datestart 选项与--datestop选项，指定具体的日期范围，如下。

```
#iptables -t filter -I OUTPUT -p tcp --dport 80 -m time --datestart 2017-12-24 --datestop 2017-12-27 -j REJECT
#
```

上图中指定的日期范围为2017年12月24日到2017年12月27日

上述选项中，--monthdays与--weekdays可以使用“!”取反，其他选项不能取反。

connlimit扩展模块

使用connlimit扩展模块，可以限制每个IP地址同时链接到server端的链接数量，注意：我们不用指定IP，其默认就是针对“每个客户端IP”，即对单IP的并发连接数

比如，我们想要限制，每个IP地址最多只能占用两个ssh链接远程到server端，我们则可以进行如下限制。

```
#iptables -I INPUT -p tcp --dport 22 -m connlimit --connlimit-above 2 -j REJECT
#
```

上例中，使用“-m connlimit”指定使用connlimit扩展，使用“--connlimit-above 2”表示限制每个IP的链接数量上限为2，再配合-p tcp --dport 22，即表示限制ssh并发链接数量不能高于2。

centos6中，我们可以对--connlimit-above选项进行取反，没错，老规矩，使用“!”对此条件进行取反，示例如下

```
#iptables -I INPUT -p tcp --dport 22 -m connlimit ! --connlimit-above 2 -j ACCEPT
#
```

上例表示，每个客户端IP的ssh链接数量只要不超过两个，则允许链接。

但是聪明如你一定想到了，上例的规则并**不能表示**：每个客户端IP的ssh链接数量超过两个则拒绝链接（与前文中的举例原理相同，此处不再赘述，如果你不明白，见文章）。也就是说，即使我们配置了上例中的规则，也不能达到“限制”的目的，所以我们通常并不会对此选项取反，因为既然使用了此选项，我们的目的通常就是“限制”。centos7中iptables为我们提供了一个新的选项，--connlimit-upto，这个选项的含义与“! --connlimit-above”的含义相同，即链接数量未达到指定的连接数量之所述，--connlimit-upto选项也不常用。

刚才说过，--connlimit-above默认表示限制“每个IP”的连接数量，其实，我们还可以配合--connlimit-mask选项，去限制“某类网段”的连接数量，示例如下：

（注：下例需要一定的网络知识基础，如果你还不了解它们，可以选择先跳过此选项或者先去学习部分的网络知识）

```
#iptables -I INPUT -p tcp --dport 22 -m connlimit --connlimit-above 2 --connlimit-mask 24 -j REJECT
```

上例中，“--connlimit-mask 24”表示某个C类网段，没错，mask为掩码之意，所以将24转换成点分十进制就表示255.255.255.0，所以，上图示例的规则表示，一4个IP的C类网络中，同时最多只能有2个ssh客户端连接到当前服务器，看来资源很紧张啊！254个IP才有2个名额，如果一个IP同时把两个连接名额都占用了，那么再连一个连接名额都没有了，那么，我们再看看下例，是不是就好多了。

```
#iptables -I INPUT -p tcp --dport 22 -m connlimit --connlimit-above 10 --connlimit-mask 27 -j REJECT
```

上例中，“--connlimit-mask 27”表示某个C类网段，通过计算后可以得知，这个网段中最多只能有30台机器（30个IP），这30个IP地址最多只能有10个ssh连接同时端，是不是比刚才的设置大方多了，当然，这样并不能避免某个IP占用所有连接的情况发生，假设，报文来自192.168.1.40这个IP，按照掩码为27进行计算，这个IP属于192.168.1.32/27网段，如果192.168.1.40同时占用了10个ssh连接，那么当192.168.1.51这个IP向服务端发起ssh连接请求时，同样会被拒绝，因为192.168.1.51这个IP按照行计算，也是属于192.168.1.32/27网段，所以他们共享这10个连接名额。

聪明如你一定明白了，在不使用--connlimit-mask的情况下，连接数量的限制是针对“每个IP”而言的，当使用了--connlimit-mask选项以后，则可以针对“某类IP段的IP”进行连接数量的限制，这样就能够灵活许多，不是吗？

limit扩展模块

刚才认识了connlimit模块，现在来认识一下limit模块。

connlimit模块是对连接数量进行限制的，limit模块是对“报文到达速率”进行限制的。

用大白话说就是，如果我想要限制单位时间内流入的包的数量，就能用limit模块。

我们可以以秒为单位进行限制，也可以以分钟、小时、天作为单位进行限制。

比如，限制每秒中最多流入3个包，或者限制每分钟最多流入30个包，都可以。

那么，我们来看一个最简单的示例，假设，我们想要限制，外部主机对本机进行ping操作时，本机最多每6秒中放行一个ping包，那么，我们可以进行如下设置（注下设置有可能无法实现限制功能，请看完后面的内容）

```
[www.zsythink.net]#iptables -F
[www.zsythink.net]#iptables -t filter -I INPUT -p icmp -m limit --limit 10/minute -j ACCEPT
```

上例中，“-p icmp”表示我们针对ping请求添加了一条规则（ping使用icmp协议），“-m limit”表示使用limit模块，“--limit 10/minute -j ACCEPT”表示每分钟最多放行10个包，就相当于每6秒钟最多放行一个包，换句话说，就是每过6秒钟放行一个包，那么配置完上述规则后，我们在另外一台机器上对当前机器进行ping操作，看看是否达到目的，如下图所示。

```
[www.zsythink.net]# ping 192.168.1.126
PING 192.168.1.126 (192.168.1.126) 56(84) bytes of data.
64 bytes from 192.168.1.126: icmp_seq=1 ttl=64 time=1.48 ms
64 bytes from 192.168.1.126: icmp_seq=2 ttl=64 time=0.307 ms
64 bytes from 192.168.1.126: icmp_seq=3 ttl=64 time=0.398 ms
64 bytes from 192.168.1.126: icmp_seq=4 ttl=64 time=0.674 ms
64 bytes from 192.168.1.126: icmp_seq=5 ttl=64 time=0.359 ms
64 bytes from 192.168.1.126: icmp_seq=6 ttl=64 time=0.319 ms
64 bytes from 192.168.1.126: icmp_seq=7 ttl=64 time=0.422 ms
64 bytes from 192.168.1.126: icmp_seq=8 zsythink.net 未双印博客
```

我们发现，刚才配置规则并没有如我们想象中的一样，ping请求的响应速率完全没有发生任何变化，为什么呢？我们一起来分析一下。

我们再来回顾一下刚才配置的规则。

```
[www.zsythink.net]#iptables -F
[www.zsythink.net]#iptables -t filter -I INPUT -p icmp -m limit --limit 10/minute -j ACCEPT
```

其实，我们可以把上图中的规则理解为如下含义。

每6秒放行一个包，那么iptables就会计时，每6秒一个轮次，到第6秒时，达到的报文就会匹配到对应的规则，执行对应的动作，而上图中的动作是ACCEPT。

那么在第6秒之前到达的包，则无法被上述规则匹配到。

之前总结过，报文会匹配链中的每一条规则，如果没有任何一条规则能够匹配到，则匹配默认动作（链的默认策略）。

既然第6秒之前的包没有被上述规则匹配到，而我们又没有在INPUT链中配置其他规则，所以，第6秒之前的包肯定会被默认策略匹配到，那么我们看看默认策略是什


```
[www.zsythink.net]#iptables -nL INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source                destination          limit: avg 10/min burst 5
ACCEPT     icmp -- 0.0.0.0/0              0.0.0.0/0
[www.zsythink.net]#
```

现在再想想，我想你应该明白为什么刚才的ping的响应速率没有变化了。

因为，上例中，第六秒的报文的确被对应的规则匹配到了，于是执行了“放行”操作，第6秒之前的报文没有被上图中配置的规则匹配到，但是被默认策略匹配到了，动作也是ACCEPT，所以，相当于所有的ping报文都被放行了，怪不得与没有配置规则时的速率一毛一样了。

那么，知错就改，聪明如你一定想到了，我们可以修改INPUT链的默认策略，或者在上例限制规则的后面再加入一条规则，将“漏网之鱼”匹配到即可，示例如下。

```
[www.zsythink.net]#iptables -F
[www.zsythink.net]#iptables -t filter -I INPUT -p icmp -m limit --limit 10/minute -j ACCEPT
[www.zsythink.net]#iptables -t filter -A INPUT -p icmp -j REJECT
[www.zsythink.net]#iptables -nL INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source                destination          limit: avg 10/min burst 5
ACCEPT     icmp -- 0.0.0.0/0              0.0.0.0/0
REJECT     icmp -- 0.0.0.0/0              0.0.0.0/0          reject-with icmp-port-unreachable
[www.zsythink.net]#
```

如上图所示，第一条规则表示每分钟最多放行10个icmp包，也就是6秒放行一个，第6秒的icmp包会被上例中的第一条规则匹配到，第6秒之前的包则不会被第一条规则匹配到，于是被后面的拒绝规则匹配到了，那么，此刻，我们再来试试，看看ping的报文放行速率有没有发生改变。

如下图所示

```
[www.zsythink.net]# ping 192.168.1.126
PING 192.168.1.126 (192.168.1.126) 56(84) bytes of data.
64 bytes from 192.168.1.126: icmp_seq=1 ttl=64 time=1.48 ms
64 bytes from 192.168.1.126: icmp_seq=2 ttl=64 time=0.483 ms
64 bytes from 192.168.1.126: icmp_seq=3 ttl=64 time=0.448 ms
64 bytes from 192.168.1.126: icmp_seq=4 ttl=64 time=0.382 ms
64 bytes from 192.168.1.126: icmp_seq=5 ttl=64 time=0.462 ms
From 192.168.1.126 icmp_seq=6 Destination Port Unreachable
64 bytes from 192.168.1.126: icmp_seq=7 ttl=64 time=0.386 ms
From 192.168.1.126 icmp_seq=8 Destination Port Unreachable
From 192.168.1.126 icmp_seq=9 Destination Port Unreachable
From 192.168.1.126 icmp_seq=10 Destination Port Unreachable
From 192.168.1.126 icmp_seq=11 Destination Port Unreachable
From 192.168.1.126 icmp_seq=12 Destination Port Unreachable
64 bytes from 192.168.1.126: icmp_seq=13 ttl=64 time=0.340 ms
From 192.168.1.126 icmp_seq=14 Destination Port Unreachable
From 192.168.1.126 icmp_seq=15 Destination Port Unreachable
From 192.168.1.126 icmp_seq=16 Destination Port Unreachable
From 192.168.1.126 icmp_seq=17 Destination Port Unreachable
From 192.168.1.126 icmp_seq=18 Destination Port Unreachable
64 bytes from 192.168.1.126: icmp_seq=19
```

刚开始还真吓我一跳，难道配置的规则还是有问题？

结果发现，只有前5个ping包没有受到限制，之后的ping包已经开始受到了规则的限制了。

从上图可以看出，除了前5个ping包以外，之后的ping包差不多每6秒才能ping通一次，看来，之后的ping包已经受到了规则的控制，被限制了流入防火墙的速率了，个ping包是什么鬼？为什么它们不受规则限制呢？其实，这个现象正好引出另一个话题，出现上图中的情况，是因为另一个选项：“--limit-burst”

limit-burst选项是干什么用的呢？我们先不准确的大白话描述一遍，“--limit-burst”可以指定“空闲时可放行的包的数量”，其实，这样说并不准确，但是我们可以理解，在不使用“--limit-burst”选项明确指定放行包的数量时，默认值为5，所以，才会出现上图中的情况，前5个ping包并没有受到任何速率限制，之后的包才受限制。

如果想要彻底了解limit模块的工作原理，我们需要先了解一下“令牌桶”算法，因为limit模块使用了令牌桶算法。

我们可以这样想象，有一个木桶，木桶里面放了5块令牌，而且这个木桶最多也只能放下5块令牌，所有报文如果想要出入关，都必须要有木桶中的令牌才行，这神奇的功能，就是每隔6秒钟会生成一块新的令牌，如果此时，木桶中的令牌不足5块，那么新生成的令牌就存放在木桶中，如果木桶中已经存在5块令牌，新生成的放了，只能溢出木桶（令牌被丢弃），如果此时有5个报文想要入关，那么这5个报文就去木桶里找令牌，正好一人一个，于是他们5个手持令牌，快乐的入关了，此再有报文想要入关，已经没有对应的令牌可以使用了，但是，过了6秒钟，新的令牌生成了，此刻，正好来了一个报文想要入关，于是，这个报文拿起这个令牌，就个报文之后，如果很长一段时间内没有新的报文想要入关，木桶中的令牌又会慢慢的积攒了起来，直到达到5个令牌，并且一直保持着5个令牌，直到有人需要使用这就是令牌桶算法的大致逻辑。

那么，就拿刚才的“令牌桶”理论类比我们的命令，“--limit”选项就是用于指定“多长时间生成一个新令牌的”，“--limit-burst”选项就是用于指定“木桶中最多存放几个在，你明白了吗？？示例如下

```
[www.zsythink.net]#iptables -F
[www.zsythink.net]#iptables -t filter -I INPUT -p icmp -m limit --limit-burst 3 --limit 10/minute -j ACCEPT
[www.zsythink.net]#iptables -t filter -A INPUT -p icmp -j REJECT
[www.zsythink.net]#
```

上例表示，令牌桶中最多能存放3个令牌，每分钟生成10个令牌（即6秒钟生成一个令牌）。

之前说过，使用“--limit”选项时，可以选择的时间单位有多种，如下

/second

