**一：查看帮助选项[#](https://www.cnblogs.com/jiujuan/p/9017495.html" \l "3215209307)**

[复制代码](javascript:void(0);)

tcpdump --help

Usage: tcpdump [-aAbdDefhHIJKlLnNOpqStuUvxX#] [ -B size ] [ -c count ]

[ -C file\_size ] [ -E algo:secret ] [ -F file ] [ -G seconds ]

[ -i interface ] [ -j tstamptype ] [ -M secret ] [ --number ]

[ -Q|-P in|out|inout ]

[ -r file ] [ -s snaplen ] [ --time-stamp-precision precision ]

[ --immediate-mode ] [ -T type ] [ --version ] [ -V file ]

[ -w file ] [ -W filecount ] [ -y datalinktype ] [ -z postrotate-command ]

[ -Z user ] [ expression ]

[复制代码](javascript:void(0);)

选项的解释:

[复制代码](javascript:void(0);)

-a：尝试将网络和广播地址转换成名称；

-c<数据包数目>：收到指定的数据包数目后，就停止进行倾倒操作；

-d：把编译过的数据包编码转换成可阅读的格式，并倾倒到标准输出；

-dd：把编译过的数据包编码转换成C语言的格式，并倾倒到标准输出；

-ddd：把编译过的数据包编码转换成十进制数字的格式，并倾倒到标准输出；

-e：在每列倾倒资料上显示连接层级的文件头；

-f：用数字显示网际网络地址；

-F<表达文件>：指定内含表达方式的文件；

-i<网络界面>：使用指定的网络截面送出数据包；

-l：使用标准输出列的缓冲区；

-n：不把主机的网络地址转换成名字；

-N：不列出域名；

-O：不将数据包编码最佳化；

-p：不让网络界面进入混杂模式；

-q ：快速输出，仅列出少数的传输协议信息；

-r<数据包文件>：从指定的文件读取数据包数据；

-s<数据包大小>：设置每个数据包的大小；

-S：用绝对而非相对数值列出TCP关联数；

-t：在每列倾倒资料上不显示时间戳记；

-tt： 在每列倾倒资料上显示未经格式化的时间戳记；

-T<数据包类型>：强制将表达方式所指定的数据包转译成设置的数据包类型；

-v：详细显示指令执行过程；

-vv：更详细显示指令执行过程；

-x：用十六进制字码列出数据包资料；

-w<数据包文件>：把数据包数据写入指定的文件。

[复制代码](javascript:void(0);)

注意：查看更多的信息，可以用命令：man tcpdump  
或者网址：https://www.tcpdump.org/tcpdump\_man.html

**二：用法[#](https://www.cnblogs.com/jiujuan/p/9017495.html" \l "3855674468)**

1：直接启动 tcpdump 将监视第一个网络接口所有流过的数据包

tcpdump

2：监控某一网络接口的数据包

tcpdump -i enp0s3

3：过滤主机

3.1 抓取所有经过enp0s3，目的或源地址是 192.168.1.101 的网络数据

tcpdump -i enp0s3 host 192.168.1.101

 3.2 指定源地址

tcpdump -i enp0s3 src host 192.168.1.101

3.3 指定目的地址

tcpdump -i enp0s3 dst host 192.168.1.101

3.4 截获主机192.168.1.101 和主机192.168.1.102 或192.168.1.103的通信

tcpdump -i enp0s3 host 192.168.1.101 and \(192.168.1.102 or 192.168.1.103 \)

3.5 如果想要获取主机192.168.1.101除了和主机192.168.1.102之外所有主机通信的ip包，使用命令：

tcpdump ip host 192.168.1.101 and !192.168.1.102

4：过滤端口

4.1 抓取所有经过 enp0s3，目的或源端口是22的网络数据

tcpdump -i enp0s3 port 22

4.2 指定源端口

|  |  |
| --- | --- |
| 1 | tcpdump -i enp0s3 src port 22 |

4.3 指定目的端口

tcpdump -i enp0s3 dst port 22

5：网络过滤

tcpdump -i enp0s3 net 192.168

tcpdump -i enp0s3 src net 192.168

tcpdump -i enp0s3 dst net 192.168

6：协议过滤

tcpdump -i enp0s3 arp

tcpdump -i enp0s3 ip

tcpdump -i enp0s3 tcp

tcpdump -i enp0s3 udp

tcpdump -i enp0s3 icmp

7：常用表达式

非 : ! or "not" (without the quotes)  
且 : && or "and"  
或 : || or "or"

7.1： 抓取目的地址是192.168.1.254或192.168.1.200端口是80的TCP数据

tcpdump '((tcp) and (port 80) and ((dst host 192.168.1.254) or (dst host 192.168.1.200)))'

当然上也可以像之前的加上指定网卡 -i enp0s3  
tcpdump -i enp0s3 '((tcp) and (port 80) and ((dst host 192.168.1.254) or (dst host 192.168.1.200)))'

7.2： 抓取目标MAC地址是00:01:02:03:04:05的ICMP数据

tcpdump '((icmp) and ((ether dst host 00:01:02:03:04:05)))'

可以加上具体网卡

7.3：抓取目的网络是192.168，但目的主机不是192.168.1.200的TCP数据

tcpdump '((tcp) and ((dst net 192.168) and (not dst host 192.168.1.200)))'

**三：高级过滤包头[#](https://www.cnblogs.com/jiujuan/p/9017495.html" \l "84167390)**

当我们继续之前，必须了解tcp/ip包头的头部信息

proto[x:y] : 过滤从x字节开始的y字节数。比如ip[2:2]过滤出3、4字节（第一字节从0开始排）

proto[x:y] & z = 0 : proto[x:y]和z的与操作为0

proto[x:y] & z !=0 : proto[x:y]和z的与操作不为0

proto[x:y] & z = z : proto[x:y]和z的与操作为z

proto[x:y] = z : proto[x:y]等于z

操作符：

[复制代码](javascript:void(0);)

> : greater 大于

< : lower 小于

>= : greater or equal 大于或者等于

<= : lower or equal 小于或者等于

= : equal 等于

!= : different 不等于

[复制代码](javascript:void(0);)

第一次在这地方看见这个你可能不是很清楚  
当然，在深入理解过滤头部包，首先要了解协议头是很重要的

1：IP头部：

[复制代码](javascript:void(0);)

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|Version| IHL |Type of Service| Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Identification |Flags| Fragment Offset |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Time to Live | Protocol | Header Checksum |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Source Address |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Destination Address |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Options | Padding | <-- optional

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| DATA ... |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

[复制代码](javascript:void(0);)

我们只考虑IPv4协议

2：IP 选项设置

“一般”的IP头是20字节，但IP头有选项设置，不能直接从偏移21字节处读取数据。IP头有个长度字段可以知道头长度是否大于20字节。

+-+-+-+-+-+-+-+-+

|Version| IHL |

+-+-+-+-+-+-+-+-+

通常第一个字节的二进制值是：01000101，分成两个部分：  
0100 = 4 表示IP版本 0101 = 5 表示IP头32 bit的块数，5 x 32 bits = 160 bits or 20 bytes  
如果第一字节第二部分的值大于5，那么表示头有IP选项。

下面介绍两种过滤方法（第一种方法比较操蛋，可忽略）

2.1. 比较第一字节的值是否大于01000101，这可以判断IPv4带IP选项的数据和IPv6的数据。  
01000101十进制等于69，计算方法如下（小提示：用计算器更方便）

[复制代码](javascript:void(0);)

0 : 0 \

1 : 2^6 = 64 \ 第一部分 (IP版本)

0 : 0 /

0 : 0 /

-

0 : 0 \

1 : 2^2 = 4 \ 第二部分 (头长度)

0 : 0 /

1 : 2^0 = 1 /  
  
64 + 4 + 1 = 69

[复制代码](javascript:void(0);)

如果设置了IP选项，那么第一自己是01000110（十进制70），过滤规则：  
tcpdump 'ip[0] > 69'  
当然可以加上网卡选项：-i enp0s3

IPv6的数据也可以匹配，第二种方法  
2.2 位操作

[复制代码](javascript:void(0);)

0100 0101 : 第一字节的二进制

0000 1111 : 与操作

<=========

0000 0101 : 结果

正确的过滤方法

tcpdump 'ip[0] & 15 > 5'

或者

tcpdump 'ip[0] & 0x0f > 5'

[复制代码](javascript:void(0);)

我用了16进制掩码.  
That's rather simple, if you want to:  
- keep the last 4 bits intact, use 0xf (binary 00001111)  
- keep the first 4 bits intact, use 0xf0 (binary 11110000)

2.3 分片标记 -Exercise: Is DF bit (don't fragment) set?

当发送端的MTU大于到目的路径链路上的MTU时就会被分片  
分片信息在IP头的第七和第八字节：

[复制代码](javascript:void(0);)

Bit 0: 保留，必须是0

Bit 1: (DF) 0 = 可能分片, 1 = 不分片

Bit 2: (MF) 0 = 最后的分片, 1 = 还有分片

Fragment Offset字段只有在分片的时候才使用。

要抓带DF位标记的不分片的包，第七字节的值应该是：

01000000 = 64

tcpdump 'ip[6] = 64'

[复制代码](javascript:void(0);)

2.4 抓分片包

a：匹配MF，分片包

tcpdump 'ip[6] = 32'

b：匹配分片和最后分片

tcpdump '((ip[6:2] > 0) and (not ip[6] = 64))'

测试分片可以用下面命令：  
ping -M want -s 3000 192.168.1.101

2.5 匹配小于ttl的数据报

TTL字段在第九字节，并且正好是完整的一个字节，TTL最大值是255，二进制为11111111。

可以来验证下，我们试着制定一个特需的ttl长度为 256  
$ ping -M want -s 3000 -t 256 192.168.1.200  
ping: ttl 256 out of range

TTL 字段：

+-+-+-+-+-+-+-+-+

| Time to Live |

+-+-+-+-+-+-+-+-+

在网关可以用下面的命令看看网络中谁在使用traceroute  
tcpdump 'ip[8] < 5'

3： 更多的过滤

tcp报文的基本结构

TCP 头

[复制代码](javascript:void(0);)

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Source Port | Destination Port |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Sequence Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Acknowledgment Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Data | |C|E|U|A|P|R|S|F| |

| Offset| Res. |W|C|R|C|S|S|Y|I| Window |

| | |R|E|G|K|H|T|N|N| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Checksum | Urgent Pointer |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Options | Padding |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| data |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

[复制代码](javascript:void(0);)

3.1 抓取源端口大于1024的TCP数据包

tcpdump 'tcp[0:2] > 1024'

or

tcpdump 'tcp src portrange 1025-65535'

3.2 匹配TCP数据包的特殊标记

TCP标记定义在TCP头的第十四个字节

+-+-+-+-+-+-+-+-+

|C|E|U|A|P|R|S|F|

|W|C|R|C|S|S|Y|I|

|R|E|G|K|H|T|N|N|

+-+-+-+-+-+-+-+-+

在TCP 3次握手中，两个主机是如何交换数据  
1、源端发送 SYN  
2、目标端口应答 SYN,ACK  
3、源端发送 ACK

- 只抓取SYN包，第十四字节是二进制的00000010，也就是十进制的2

tcpdump 'tcp[13] = 2'

- 抓取 SYN,ACK  （00010010 or 18）

tcpdump 'tcp[13] = 18'

- 抓取SYN或者SYN-ACK

tcpdump 'tcp[13] & 2 = 2'

我们使用了掩码，它会返回任何事情，当ACK是二进制设置时候  
让我们看看下面的例子（SYN-ACK)

00010010 : SYN-ACK packet

00000010 : mask (2 in decimal)

==========

00000010 : result (2 in decimal)

- 抓取PSH-ACK

tcpdump 'tcp[13] = 24'

- 抓所有包含FIN标记的包（FIN通常和ACK一起，表示幽会完了，回头见）

tcpdump 'tcp[13] & 1 = 1'

- 抓取RST

tcpdump 'tcp[13] & 4 = 4'

TCP标记值：

tcp-fin, tcp-syn, tcp-rst, tcp-push, tcp-push, tcp-ack, tcp-urg

- 抓取TCP标志位

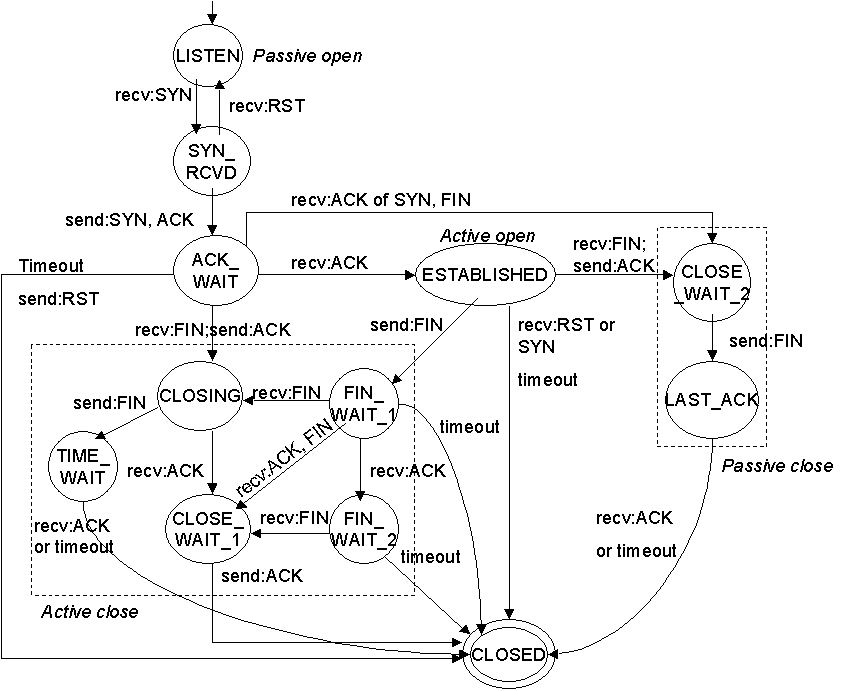
实际上有一个很简单的方法过滤 flags（man pcap-filter and look for tcpflags）

tcpdump 'tcp[tcpflags] == tcp-ack'

- 抓取所有的包，用TCP-SYN 或者 TCP-FIN 设置

tcpdump 'tcp[tcpflags] & (tcp-syn|tcp-fin) != 0

下图表示了TCP各状态转换的标记

[](https://images2018.cnblogs.com/blog/650581/201805/650581-20180510003225443-782852645.jpg)

tcpdump 提供了常用的字段偏移名字：

icmptype (ICMP类型字段)

icmpcode (ICMP符号字段)

tcpflags (TCP标记字段)

ICMP类型值有：  
icmp-echoreply, icmp-unreach, icmp-sourcequench, icmp-redirect, icmp-echo, icmp-routeradvert, icmp-routersolicit,  
icmp-timxceed, icmp-paramprob, icmp-tstamp, icmp-tstampreply, icmp-ireq, icmp-ireqreply, icmp-maskreq, icmp-maskreply

4： SMTP 数据过滤

我们将弄一个匹配任意包的过滤，这个包包括 “MAIL”  
你可以用网址 http://www.easycalculation.com/ascii-hex.php 把ASCII转化为 十六进制， 也可以用python来转化  
$ python -c 'print "MAIL".encode("hex")'  
4d41494c

所以 “MAIL” 的十六进制是：0x4d41494c  
那么规则就是

tcpdump '((port 25) and (tcp[20:4] = 0x4d41494c))'

这是一个包的例子

[复制代码](javascript:void(0);)

# tshark -V -i eth0 '((port 25) and (tcp[20:4] = 0x4d41494c))'

Capturing on eth0

Frame 1 (92 bytes on wire, 92 bytes captured)

Arrival Time: Sep 25, 2007 00:06:10.875424000

[Time delta from previous packet: 0.000000000 seconds]

[Time since reference or first frame: 0.000000000 seconds]

Frame Number: 1

Packet Length: 92 bytes

Capture Length: 92 bytes

[Frame is marked: False]

[Protocols in frame: eth:ip:tcp:smtp]

Ethernet II, Src: Cisco\_X (00:11:5c:X), Dst: 3Com\_X (00:04:75:X)

Destination: 3Com\_X (00:04:75:X)

Address: 3Com\_X (00:04:75:X)

.... ...0 .... .... .... .... = IG bit: Individual address (unicast)

.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)

Source: Cisco\_X (00:11:5c:X)

Address: Cisco\_X (00:11:5c:X)

.... ...0 .... .... .... .... = IG bit: Individual address (unicast)

.... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)

Type: IP (0x0800)

Internet Protocol, Src: 62.163.X (62.163.X), Dst: 192.168.X (192.168.X)

Version: 4

Header length: 20 bytes

Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)

0000 00.. = Differentiated Services Codepoint: Default (0x00)

.... ..0. = ECN-Capable Transport (ECT): 0

.... ...0 = ECN-CE: 0

Total Length: 78

Identification: 0x4078 (16504)

Flags: 0x04 (Don't Fragment)

0... = Reserved bit: Not set

.1.. = Don't fragment: Set

..0. = More fragments: Not set

Fragment offset: 0

Time to live: 118

Protocol: TCP (0x06)

Header checksum: 0x08cb [correct]

[Good: True]

[Bad : False]

Source: 62.163.X (62.163.X)

Destination: 192.168.X (192.168.XX)

Transmission Control Protocol, Src Port: 4760 (4760), Dst Port: smtp (25), Seq: 0, Ack: 0, Len: 38

Source port: 4760 (4760)

Destination port: smtp (25)

Sequence number: 0 (relative sequence number)

[Next sequence number: 38 (relative sequence number)]

Acknowledgement number: 0 (relative ack number)

Header length: 20 bytes

Flags: 0x18 (PSH, ACK)

0... .... = Congestion Window Reduced (CWR): Not set

.0.. .... = ECN-Echo: Not set

..0. .... = Urgent: Not set

...1 .... = Acknowledgment: Set

.... 1... = Push: Set

.... .0.. = Reset: Not set

.... ..0. = Syn: Not set

.... ...0 = Fin: Not set

Window size: 17375

Checksum: 0x6320 [correct]

[Good Checksum: True]

[Bad Checksum: False]

Simple Mail Transfer Protocol

Command: MAIL FROM:<wguthrie\_at\_mysickworld--dot--com>\r\n

Command: MAIL

Request parameter: FROM:<wguthrie\_at\_mysickworld--dot--com>

[复制代码](javascript:void(0);)

5： HTTP数据过滤

http请求开始格式  
GET / HTTP/1.1\r\n (16 bytes counting the carriage return but not the backslashes !)

“GET ” 十六进制是 47455420

tcpdump 'tcp[32:4] = 0x47455420'

HTTP数据（从man tcpdump 看到的例子）

打印所有源或目的端口是80, 网络层协议为IPv4, 并且含有数据,而不是SYN,FIN以及ACK-only等不含数据的数据包.(ipv6的版本的表达式可做练习)

tcpdump 'tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)'

[复制代码](javascript:void(0);)

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

ip[2:2] = | Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+

ip[0] = |Version| IHL |

+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+

ip[0]&0xf = |# # # #| IHL | <-- that's right, we masked the version bits

+-+-+-+-+-+-+-+-+ with 0xf or 00001111 in binary

+-+-+-+-+

| Data |

tcp[12] = | Offset|

| |

+-+-+-+-+

[复制代码](javascript:void(0);)

6： SSH过滤

我们看看ssh server  
OpenSSH 常常应答一些内容，比如"SSH-2.0-OpenSSH\_3.6.1p2"， 这第一个4 bytes (SSH-)的十六进制值是 0x5353482D

tcpdump 'tcp[(tcp[12]>>2):4] = 0x5353482D'

如果我们想要找到老版本的OpenSSH的任意链接  
这时候OpenSSH服务器应答的内容：比如 “SSH-1.99..”

tcpdump '(tcp[(tcp[12]>>2):4] = 0x5353482D) and (tcp[((tcp[12]>>2)+4):2] = 0x312E)'

7： UDP头

[复制代码](javascript:void(0);)

0 7 8 15 16 23 24 31

+--------+--------+--------+--------+

| Source | Destination |

| Port | Port |

+--------+--------+--------+--------+

| | |

| Length | Checksum |

+--------+--------+--------+--------+

| |

| DATA ... |

+-----------------------------------+

[复制代码](javascript:void(0);)

如果我们想要过滤，我们可以用下面的方法

tcpdump udp dst port 53

8：ICMP头

这里本来显示一张图片，可是原文图片没有正确显示，如实我找了一个图，如下

[复制代码](javascript:void(0);)

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Type | Code | Checksum |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| unused |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Internet Header + 64 bits of Original Data Datagram |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

[复制代码](javascript:void(0);)

在ICMP报文中，我们经常过滤 type（1 byte）和code（1 byte）

下图是ICMP报文中 Type ，我们经常用到的几个值：

[复制代码](javascript:void(0);)

0 Echo Reply [RFC792] 回显应答报文

3 Destination Unreachable [RFC792] 目的不可达

4 Source Quench [RFC792] 源冷却报文

5 Redirect [RFC792] 重定向报文

8 Echo [RFC792] 请求回显报文

11 Time Exceeded [RFC792] 超时报文

[复制代码](javascript:void(0);)

如果我们要过滤报文 type = 4

tcpdump 'icmp[0] = 4'

如我我们仅仅是要找到ICMP 的回显 应答报文，同时 ID是500。

tcpdump -i eth0 '(icmp[0] = 0) and (icmp[4:2] = 0x1f4)'

References  
tcpdump man page : http://www.tcpdump.org/tcpdump\_man.html  
Conversions: http://easycalculation.com/hex-converter.php  
Filtering HTTP requests: http://www.wireshark.org/tools/string-cf.html  
Filtering data regardless of TCP options: http://www.wireshark.org/lists/wireshark-users/201003/msg00024.html

转自：https://blog.wains.be/2007/2007-10-01-tcpdump-advanced-filters/