<http://wfeng520.blog.hexun.com.tw/47990031_d.html>

DISPLAY FILTER（顯示過濾器）

|  |
| --- |
| [wireshark display filter](http://openmaniak.com/wireshark/wireshark_use_display_filter.png) |

顯示過濾器用於查找捕捉記錄中的內容。不要將捕捉過濾器和顯示過濾器的概念混淆。參考[Wireshark過濾器](http://openmaniak.com/cn/wireshark_filters.php)中的詳細內容。  
PACKET LIST PANE（封包列表）

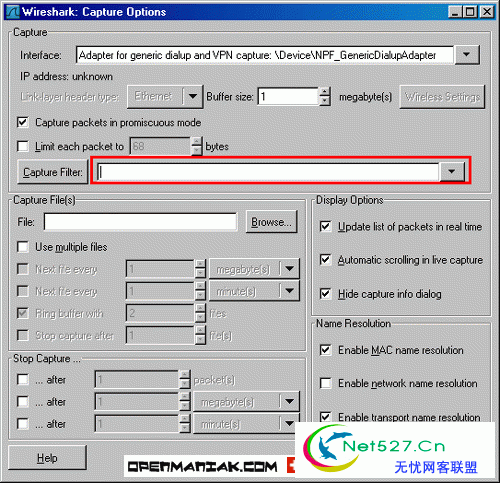
|  |
| --- |
| [wireshark packet filter pane](http://openmaniak.com/wireshark/wireshark_packet-list-pane.png) |

|  |
| --- |
| [wireshark packet filter pane](http://openmaniak.com/wireshark/wireshark_packet-list-pane2.png) |

封包列表中顯示已捕獲的封包。可看到發送或接收方的MAC/IP地址，TCP/UDP端口號，協議或者封包的內容。OSI layer 2的封包，在Source(來源)和Destination(目的地)列中看到的將是MAC地址，Port(端口)列將會為空。OSI layer 3或者更高層的封包，在Source(來源)和Destination(目的地)列中看到的將是IP地址。Port(端口)列僅會在封包屬於第4或者更高層時才會顯示。

Wireshark使用默認設置時，會得到大量冗余信息。過濾器可幫助我們在龐雜的結果中迅速找到我們需要的信息。

|  |  |
| --- | --- |
| - - | 捕捉過濾器：用於決定將什麽樣的信息記錄在捕捉結果中。需要在開始捕捉前設置。 顯示過濾器：在捕捉結果中進行詳細查找。他們可以在得到捕捉結果後隨意修改。 |

捕捉過濾器是數據經過的第一層過濾器，它用於控制捕捉數據的數量，以避免產生過大的日誌文件。顯示過濾器是一種更為強大(復雜)的過濾器。它允許您在日誌文件中迅速準確地找到所需要的記錄。兩種過濾器使用的語法是完全不同的：  
http://www.net527.cn/uploads/allimg/100105/21124J532-11.gif1. 捕捉過濾器: 捕捉過濾器的語法與Lipcap(Linux)或者Winpcap(Windows)庫開發的軟件一樣，如著名的[TCPdump](http://openmaniak.com/tcpdump.php)。捕捉過濾器須在開始捕捉前設置完畢，這一點跟顯示過濾器不同。設置捕捉過濾器的步驟是：(1) 選擇 capture -> options。(2) 填寫"capture filter"欄或者點擊"capture filter"按鈕為過濾器起一個名字並保存，以便今後的捕捉中繼續用這個過濾器。(3) 點擊開始（Start）進行捕捉。  


|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 語法： |  | **Protocol** |  | **Direction** |  | **Host(s)** |  | **Value** |  | **Logical Operations** |  | **Other expression\_r** |
| 例子： |  | tcp |  | Dst |  | 10.1.1.1 |  | 80 |  | And |  | tcp dst 10.2.2.2 3128 |

http://www.net527.cn/uploads/allimg/100105/21124MF7-14.gif**Protocol（協議）**: 可能的值: ether, fddi, ip, arp, rarp, decnet, lat, sca, moprc, mopdl, tcp and udp.  
如果沒有特別指明是什麽協議，則默認使用所有支持的協議。  
http://www.net527.cn/uploads/allimg/100105/21124MF7-14.gif**Direction（方向）**: 可能的值: src, dst, src and dst, src or dst.  
如果沒有特別指明來源或目的地，則默認使用 "src or dst" 作為關鍵字。  
例如，"host 10.2.2.2"與"src or dst host 10.2.2.2"是一樣的。  
http://www.net527.cn/uploads/allimg/100105/21124MF7-14.gif**Host(s)**: 可能的值： net, port, host, portrange.  
如果沒有指定此值，則默認使用"host"關鍵字。  
例如，"src 10.1.1.1"與"src host 10.1.1.1"相同。  
http://www.net527.cn/uploads/allimg/100105/21124MF7-14.gif**Logical Operations（邏輯運算）**: 可能的值：not, and, or.  
NOT有最高的優先級。OR和AND有相同的優先級，運算時從左至右進行。如"not tcp port 3128 and tcp port 23"與"(not tcp port 3128) and tcp port 23"相同。"not tcp port 3128 and tcp port 23"與"not (tcp port 3128 and tcp port 23)"不同。例子：

|  |
| --- |
| **tcp dst port 3128** |

顯示目的TCP端口為3128的封包。

|  |
| --- |
| **ip src host 10.1.1.1** |

顯示來源IP地址為10.1.1.1的封包。

|  |
| --- |
| **host 10.1.2.3** |

顯示目的或來源IP地址為10.1.2.3的封包。

|  |
| --- |
| **src portrange 2000-2500** |

顯示來源為UDP或TCP，並且端口號在2000至2500範圍內的封包。

|  |
| --- |
| **not imcp** |

顯示除了icmp以外的所有封包。（icmp通常被ping工具使用）

|  |
| --- |
| **src host 10.7.2.12 and not dst net 10.200.0.0/16** |

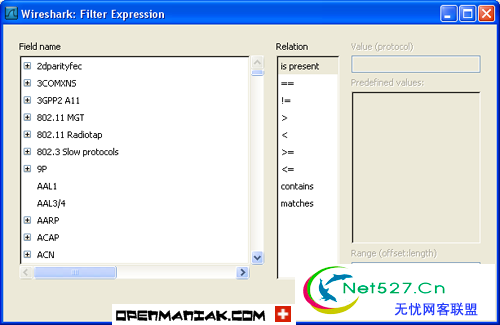
顯示來源IP地址為10.7.2.12，但目的地不是10.200.0.0/16的封包。

|  |
| --- |
| **(src host 10.4.1.12 or src net 10.6.0.0/16) and tcp dst portrange 200-10000 and dst net 10.0.0.0/8** |

顯示來源IP為10.4.1.12或來源網絡為10.6.0.0/16，目的地TCP端口號在200至10000之間，且目的位於網絡10.0.0.0/8內的封包。

註意事項：當使用關鍵字作為值時，需使用反斜杠“\”。  
"ether proto \ip" (與關鍵字"ip"相同).這樣寫將會以IP協議作為目標。  
"ip proto \icmp" (與關鍵字"icmp"相同).這樣寫將會以ping工具常用的icmp作為目標。  
可在"ip"或"ether"後面使用"multicast"及"broadcast"關鍵字。想排除廣播請求時，"no broadcast"就會非常有用。查看 [TCPdump的主頁](http://www.ethereal.com/docs/man-pages/tcpdump.8.html#_blank)以獲得更詳細的捕捉過濾器語法說明。在[Wiki Wireshark website](http://wiki.wireshark.org/CaptureFilters#_blank)上可以找到更多捕捉過濾器的例子。  
http://www.net527.cn/uploads/allimg/100105/21124J532-11.gif2. 顯示過濾器：捕捉過濾器過濾後的數據還是很復雜。可用顯示過濾器進行更細致的查找。在想修改過濾器條件時，不需重新捕捉一次。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 語法： |  | **Protocol** | **.** | **String 1** | **.** | **String 2** |  | **Comparison operator** |  | **Value** |  | **Logical Operations** |  | **Other expression\_r** |
| 例子： |  | ftp |  | passive |  | Ip |  | == |  | 10.2.3.4 |  | Xor |  | icmp.type |

http://www.net527.cn/uploads/allimg/100105/21124MF7-14.gif**Protocol（協議）**:  
可使用位於OSI模型第2至7層的協議。點擊"Expression..."按鈕後，您可以看到它們。比如：IP，TCP，DNS，SSH  
[](http://openmaniak.com/wireshark/wireshark_display_filter_1.png)  
http://www.net527.cn/uploads/allimg/100105/21124MF7-14.gif**String1, String2** (可選項):協議的子類。點擊相關父類旁的"+"號，然後選擇其子類。  
http://www.net527.cn/uploads/allimg/100105/21124MF7-14.gif**Comparison operators （比較運算符）**:可以使用6種比較運算符

|  |  |  |
| --- | --- | --- |
| 英文寫法： | C語言寫法： | 含義： |
| eq | == | 等於 |
| ne | != | 不等於 |
| gt | > | 大於 |
| lt | < | 小於 |
| ge | >= | 大於等於 |
| le | <= | 小於等於 |

http://www.net527.cn/uploads/allimg/100105/21124MF7-14.gif**Logical expression\_rs（邏輯運算符）**:

|  |  |  |
| --- | --- | --- |
| 英文寫法： | C語言寫法： | 含義： |
| and | && | 邏輯與 |
| or | || | 邏輯或 |
| xor | ^^ | 邏輯異或 |
| not | ! | 邏輯非 |

當被用在過濾器的兩個條件間，只有當且僅當其中的一個條件滿足時，才會被顯示在屏幕上。

**tcp.dstport 80 xor tcp.dstport 1025**

只有當目的TCP端口為80或者來源於端口1025(但又不能同時滿足這兩點)時的封包才會被顯示。

**snmp || dns || icmp**

顯示SNMP或DNS或ICMP封包。

|  |
| --- |
| **ip.addr == 10.1.1.1** |

顯示來源或目的IP地址為10.1.1.1的封包。

|  |
| --- |
| **ip.src != 10.1.2.3 or ip.dst != 10.4.5.6** |

顯示來源不為10.1.2.3或者目的不為10.4.5.6的封包。換句話說，顯示的封包將會為：  
(1) 來源IP：除了10.1.2.3以外任意；目的IP：任意。(2) 來源IP：任意；目的IP：除了10.4.5.6以外任意。

|  |
| --- |
| **ip.src != 10.1.2.3 and ip.dst != 10.4.5.6** |

顯示來源不為10.1.2.3並且目的IP不為10.4.5.6的封包。換句話說，顯示的封包將會為：  
來源IP：除了10.1.2.3以外任意；同時須滿足，目的IP：除了10.4.5.6以外任意

|  |  |
| --- | --- |
| **tcp.port == 25** | 顯示來源或目的TCP端口號為25的封包。 |

|  |  |
| --- | --- |
| **tcp.dstport == 25** | 顯示目的TCP端口號為25的封包。 |

|  |  |
| --- | --- |
| **tcp.flags** | 顯示包含TCP標誌的封包。 |

|  |  |
| --- | --- |
| **tcp.flags.syn == 0x02** | 顯示包含TCP SYN標誌的封包。 |

如果過濾器的語法是正確的，表達式的背景呈綠色。如果呈紅色，說明表達式有誤。

|  |  |
| --- | --- |
| wireshark display filter example | 表達式正確 |
| wireshark display filter example | 表達式錯誤 |

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Note that tcpdump only takes the first 96 bytes of data from a packet by default. If you would like to look at more, add the **-s *number*** option to the mix, where *number* is the number of bytes you want to capture. I recommend using 0 (zero) for a snaplength, which gets everything. Here's a short list of the options I use most:

* **-i any** : Listen on all interfaces just to see if you're seeing any traffic.
* **-n** : Don't resolve hostnames.
* **-nn** : Don't resolve hostnames *or* port names.
* **-X** : Show the packet's *contents* in both [hex](http://en.wikipedia.org/wiki/Hexidecimal) and [ASCII](http://en.wikipedia.org/wiki/Ascii).
* **-XX** : Same as **-X**, but also shows the ethernet header.
* **-v, -vv, -vvv** : Increase the amount of packet information you get back.
* **-c** : Only get *x* number of packets and then stop.
* **-s** : Define the *snaplength* (size) of the capture in bytes. Use -s0to get everything.
* **-S** : Print absolute sequence numbers.
* **-e** : Get the ethernet header as well.
* **-q** : Show less protocol information.
* **-E** : Decrypt IPSEC traffic by providing an encryption key.

*[ The default snaplength as of tcpdump 4.0 has changed from 68 bytes to 96 bytes. While this will give you more of a packet to see, it still won't get everything. Use-s 1514 to get full coverage ]*

Here's a capture of exactly two (-c2) [ICMP](http://en.wikipedia.org/wiki/Icmp) packets (a [ping](http://www.networksorcery.com/enp/protocol/icmp/msg8.htm) and [pong](http://www.networksorcery.com/enp/protocol/icmp/msg0.htm)) using some of the options described above. Notice how much we see about each packet.

hermes root # tcpdump -nnvXSs 0 -c2 icmp

tcpdump: listening on eth0, link-type EN10MB (Ethernet), 23:11:10.370321 IP

(tos 0x20, ttl 48, id 34859, offset 0, flags [none], length: 84)

69.254.213.43 > 72.21.34.42: icmp 64: echo request seq 0

0x0000: 4520 0054 882b 0000 3001 7cf5 45fe d52b E..T.+..0.|.E..+

0x0010: 4815 222a 0800 3530 272a 0000 25ff d744 H."\*..50'\*..%..D

0x0020: ae5e 0500 0809 0a0b 0c0d 0e0f 1011 1213 .^..............

0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223 .............!"#

0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233 $%&'()\*+,-./0123

0x0050: 3435 3637 4567

23:11:10.370344 IP (tos 0x20, ttl 64, id 35612, offset 0, flags [none],

length: 84) 72.21.34.42 > 69.254.213.43: icmp 64: echo reply seq 0

0x0000: 4520 0054 8b1c 0000 4001 6a04 4815 222a E..T....@.j.H."\*

0x0010: 45fe d52b 0000 3d30 272a 0000 25ff d744 E..+..=0'\*..%..D

0x0020: ae5e 0500 0809 0a0b 0c0d 0e0f 1011 1213 .^..............

0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223 .............!"#

0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233 $%&'()\*+,-./0123

0x0050: 3435 3637 4567

2 packets captured

2 packets received by filter

0 packets dropped by kernel

Common Syntax

Expressions allow you to trim out various types of traffic and find exactly what you're looking for. Mastering the expressions and learning to combine them creatively is what makes one truly powerful with tcpdump. There are three main types of expression: type, dir, andproto. Type options are host, net, and port. Direction is indicated by dir, and there you can have src, dst, src or dst, and src and dst. Here are a few that you should definitely be comfortable with:

* **host** // look for traffic based on IP address (also works with hostname if you're not using **-n**)

# tcpdump host 1.2.3.4

* **src**, **dst** // find traffic from only a source or destination (eliminates one side of a **host** conversation)

# tcpdump src 2.3.4.5   
# tcpdump dst 3.4.5.6

* **net** // capture an entire network using [CIDR](http://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing) notation

# tcpdump net 1.2.3.0/24

* **proto** // works for tcp, udp, and icmp. Note that you don't have to type proto

# tcpdump icmp

* **port** // see only traffic to or from a certain port

# tcpdump port 3389

* **src, dst port** // filter based on the source or destination port

# tcpdump src port 1025   
# tcpdump dst port 389

* **src/dst, port, protocol** // combine all three

# tcpdump src port 1025 and tcp   
# tcpdump udp and src port 53

You also have the option to filter by a *range* of ports instead of declaring them individually, and to only see packets that are above or below a certain size.

* **Port Ranges** // see traffic to any port in a range   
  tcpdump portrange 21-23
* **Packet Size Filter** // only see packets below or above a certain size (in bytes)   
  tcpdump less 32   
  tcpdump greater 128

*[ You can use the symbols for less than, greater than, and less than or equal/ greater than or equal signs as well. ]*

// filtering for size using symbols   
tcpdump > 32   
tcpdump <= 128

Writing to a File

tcpdump allows you to send what you're capturing to a file for later use using the **-w** option, and then to read it back using the **-r** option. This is an excellent way to capture raw traffic and then run it through various tools later.

**Capture all Port 80 Traffic to a File**

# tcpdump -s 1514 port 80 -w capture\_file

Then, at some point in the future, you can then read the traffic back in like so:

**Read Captured Traffic back into tcpdump**

# tcpdump -r capture\_file

Getting Creative

Expressions are nice, but the real magic of tcpdump comes from the ability to *combine* them in creative ways in order to isolate exactly what you're looking for. There are three ways to do combinations:

1. **AND**  *and* or &&
2. **OR**  *or* or ||
3. **EXCEPT**  *not* or !

More Examples

*# TCP traffic from 10.5.2.3 destined for port 3389*

tcpdump -nnvvS and src 10.5.2.3 and dst port 3389

*# Traffic originating from the 192.168 network headed for the 10 or 172.16 networks*

tcpdump -nvX src net 192.168.0.0/16 and dst net 10.0.0.0/8 or 172.16.0.0/16

*# Non-ICMP traffic destined for 192.168.0.2 from the 172.16 network*

tcpdump -nvvXSs 1514 dst 192.168.0.2 and src net and not icmp

*# Traffic originating from Mars or Pluto that isn't to the SSH port*

tcpdump -vv src mars and not dst port 22

Grouping

Also remember that when you're building complex queries you might have to group your options using single quotes. Single quotes are used in order to tell tcpdump to ignore certain special characters -- in this case the "( )" brackets. This same technique can be used to group using other expressions such as host, port, net, etc. Take a look at the command below:

*# Traffic that's from 10.0.2.4 AND destined for ports 3389 or 22****(incorrect)***

tcpdump src 10.0.2.4 and (dst port 3389 or 22)

If you tried to run this otherwise very useful command, you'd get an error because of the parenthesis. You can either fix this by escaping the parenthesis (putting a \ before each one), or by putting the entire command within single quotes:

*# Traffic that's from 10.0.2.4 AND destined for ports 3389 or 22****(correct)***

tcpdump 'src 10.0.2.4 and (dst port 3389 or 22)'