Wireshark 图解教程（简介、抓包、过滤器）配置

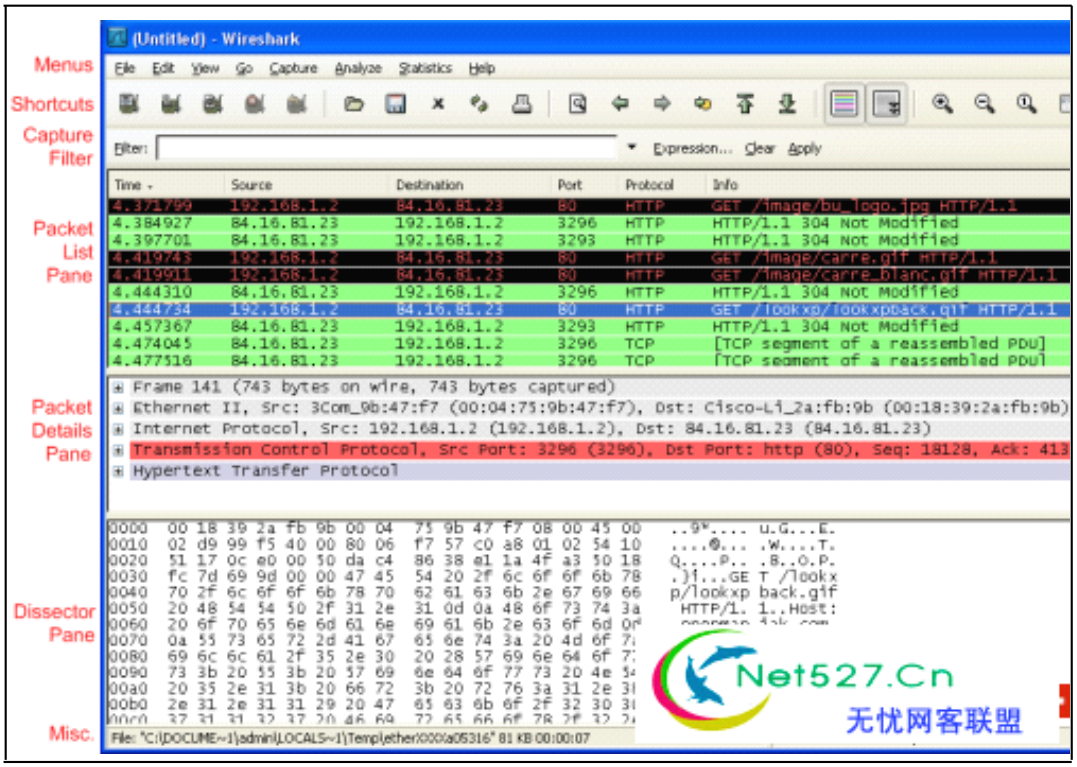
Wireshark 是世界上最流行的网络分析工具。这个强大的工具可以捕捉网络中的数据，并为用户提供关于网络和上层协议的各种信息。与很多其他网络工具一样，Wireshark 也使用 pcap network library 来进行封包捕捉。 可破解局域网内 QQ、邮箱、msn、账号等的密码 ！！

Wireshark 是世界上最流行的网络分析工具。这个强大的工具可以捕捉网络中的数据，并为用户提供关于网络和上层协议的各种信息。与很多其他网络工具一样，Wireshark 也使用 pcap network library 来进行封包捕捉。可破解局域网内 QQ、邮箱、msn、账号等的密码！！

wireshark 的原名是 Ethereal，新名字是 2006 年起用的。当时 Ethereal 的主要开发者决定离开他原来供职的公司，并继续开发这个软件。但由于 Ethereal 这个名称的使用权已经被原来那个公司注册，Wireshark 这个新名字也就应运而生了。

在成功运行 Wireshark 之后，我们就可以进入下一步，更进一步了解这个强大的工具。

下面是一张地址为 192.168.1.2 的计算机正在访问“openmaniak.com”网站时的截图。



1. MENUS（菜单）

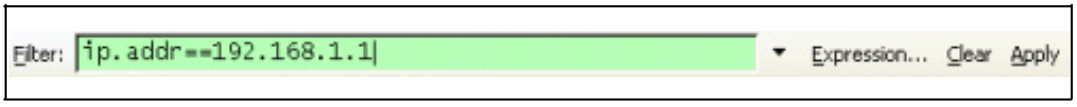
程序上方的 8 个菜单项用于对 Wireshark 进行配置：

- "File"（文件）　　打开或保存捕获的信息。
- "Edit"（编辑）　　查找或标记封包。进行全局设置。
- "View"（查看）
- "Go"（转到）　　设置 Wireshark 的视图。
- "Capture"（捕获）　　跳转到捕获的数据。
- "Analyze"（分析）　　设置捕捉过滤器并开始捕捉。
- "Statistics"（统计）　　设置分析选项。
  查看 Wireshark 的统计信息。
- "Help"（帮助）　　查看本地或者在线支持。

2. SHORTCUTS（快捷方式）

在菜单下面，是一些常用的快捷按钮。
您可以将鼠标指针移动到某个图标上以获得其功能说明。

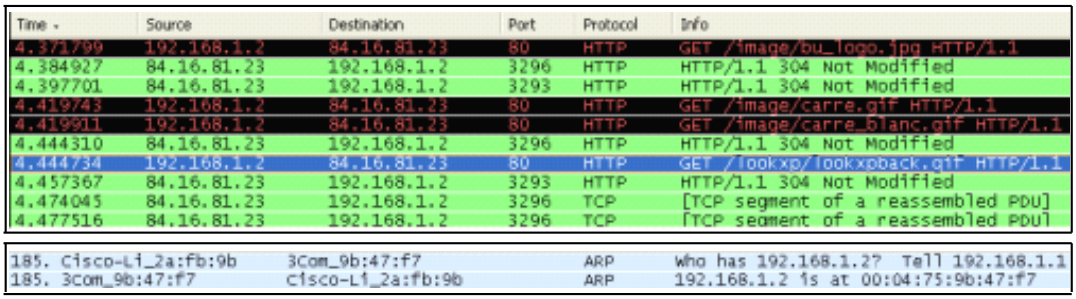
3. DISPLAY FILTER（显示过滤器）



显示过滤器用于查找捕捉记录中的内容。
请不要将捕捉过滤器和显示过滤器的概念相混淆。请参考 Wireshark 过滤器中的详细内容。

返回页面顶部

4. PACKET LIST PANE（封包列表）

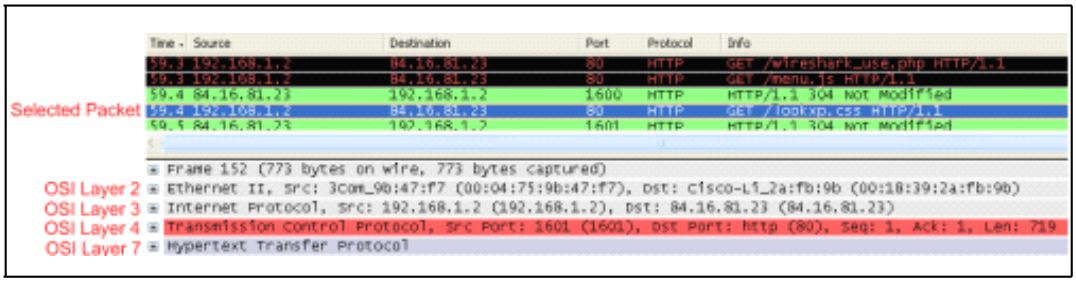


封包列表中显示所有已经捕获的封包。在这里您可以看到发送或接收方的 MAC/IP 地址，TCP/UDP 端口号，协议或者封包的内容。

如果捕获的是一个 OSI layer 2 的封包，您在 Source（来源）和 Destination（目的地）列中看到的将是 MAC 地址，当然，此时 Port（端口）列将会为空。
如果捕获的是一个 OSI layer 3 或者更高层的封包，您在 Source（来源）和 Destination（目的地）列中看到的将是 IP 地址。Port（端口）列仅会在这个封包属于第 4 或者更高层时才会显示。

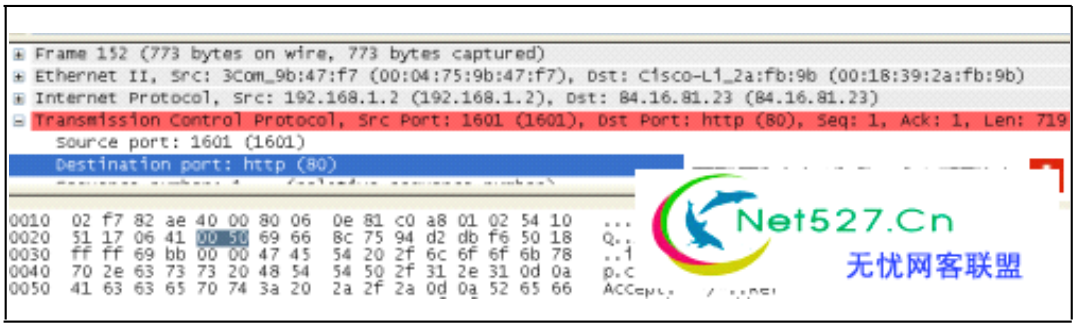
您可以在这里添加/删除列或者改变各列的颜色：
Edit menu -> Preferences

5. PACKET DETAILS PANE（封包详细信息）



这里显示的是在封包列表中被选中项目的详细信息。
信息按照不同的 OSI layer 进行了分组，您可以展开每个项目查看。下面截图中展开的是 HTTP 信息。

6. DISSECTOR PANE（16 进制数据）



“解析器”在 Wireshark 中也被叫做“16 进制数据查看面板”。这里显示的内容与“封包详细信息”中相同，只是改为以 16 进制的格式表述。
在上面的例子里，我们在“封包详细信息”中选择查看 TCP 端口（80），其对应的 16 进制数据将自动显示在下面的面板中（0050）。

7. MISCELLANOUS（杂项）

在程序的最下端，您可以获得如下信息：

- - 正在进行捕捉的网络设备。
- 捕捉是否已经开始或已经停止。
- 捕捉结果的保存位置。
- 已捕捉的数据量。
- 已捕捉封包的数量。(P)
- 显示的封包数量。(D) (经过显示过滤器过滤后仍然显示的封包)
- 被标记的封包数量。(M)

正如您在 Wireshark 教程第一部分看到的一样，安装、运行 Wireshark 并开始分析网络是非常简单的。

使用 Wireshark 时最常见的问题，是当您使用默认设置时，会得到大量冗余信息，以至于很难找到自己需要的部分。
过犹不及。

这就是为什么过滤器会如此重要。它们可以帮助我们在庞杂的结果中迅速找到我们需要的信息。

- 捕捉过滤器：用于决定将什么样的信息记录在捕捉结果中。需要在开始捕捉前设置。

- 显示过滤器：在捕捉结果中进行详细查找。他们可以在得到捕捉结果后随意修改。

那么我应该使用哪一种过滤器呢？

两种过滤器的目的是不同的。

捕捉过滤器是数据经过的第一层过滤器，它用于控制捕捉数据的数量，以避免产生过大的日志文件。
显示过滤器是一种更为强大（复杂）的过滤器。它允许您在日志文件中迅速准确地找到所需要的记录。

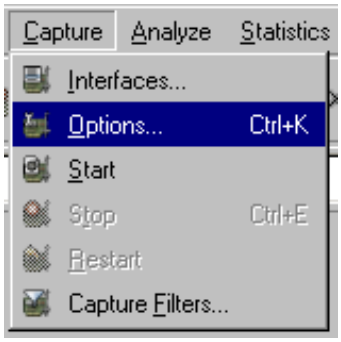
两种过滤器使用的语法是完全不同的。我们将在接下来的几页中对它们进行介绍：

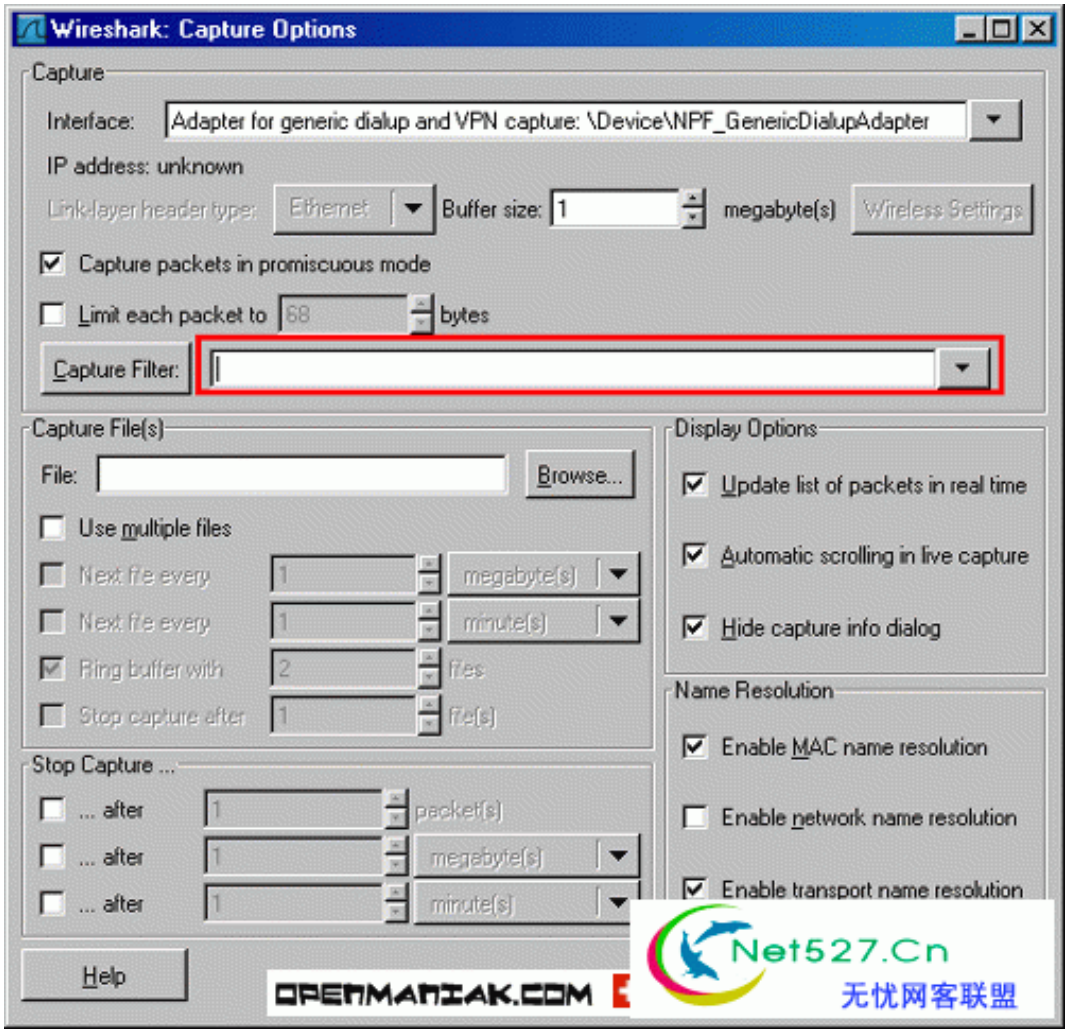
---

1. 捕捉过滤器

捕捉过滤器的语法与其它使用 Lipcap（Linux）或者 Winpcap（Windows）库开发的软件一样，比如著名的 TCPdump。捕捉过滤器必须在开始捕捉前设置完毕，这一点跟显示过滤器是不同的。

设置捕捉过滤器的步骤是：
- 选择 capture -> options。
- 填写"capture filter"栏或者点击"capture filter"按钮为您的过滤器起一个名字并保存，以便在今后的捕捉中继续使用这个过滤器。
- 点击开始（Start）进行捕捉。





语法：	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

Protocol（协议）：
可能的值：ether, fddi, ip, arp, rarp, decnet, lat, sca, moprc, mopdl, tcp and udp.

如果没有特别指明是什么协议，则默认使用所有支持的协议。

Direction（方向）：
可能的值：src, dst, src and dst, src or dst
如果没有特别指明来源或目的地，则默认使用 "src or dst" 作为关键字。
例如，"host 10.2.2.2"与"src or dst host 10.2.2.2"是一样的。

Host(s)：
可能的值： net, port, host, portrange.
如果没有指定此值，则默认使用"host"关键字。
例如，"src 10.1.1.1"与"src host 10.1.1.1"相同。

Logical Operations（逻辑运算）：

可能的值：not, and, or.
否("not")具有最高的优先级。或("or")和与("and")具有相同的优先级，运算时从左至右进行。
例如，
"not tcp port 3128 and tcp port 23"与"(not tcp port 3128) and tcp port 23"相同。

"not tcp port 3128 and tcp port 23"与"not (tcp port 3128 and tcp port 23)"不同。

例子：

tcp dst port 3128
显示目的 TCP 端口为 3128 的封包。

ip src host 10.1.1.1
显示来源 IP 地址为 10.1.1.1 的封包。

host 10.1.2.3
显示目的或来源 IP 地址为 10.1.2.3 的封包。

src portrange 2000-2500
显示来源为 UDP 或 TCP，并且端口号在 2000 至 2500 范围内的封包。

not imcp
显示除了 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
显示来源 IP 地址为 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
显示来源 IP 为 10.4.1.12 或者来源网络为 10.6.0.0/16，目的地 TCP 端口号在 200 至 10000 之间，并且目的位于网络 10.0.0.0/8 内的所有封包。

注意事项：

当使用关键字作为值时，需使用反斜杠“\”。
"ether proto \ip" (与关键字"ip"相同).
这样写将会以 IP 协议作为目标。

"ip proto \icmp" (与关键字"icmp"相同).
这样写将会以 ping 工具常用的 icmp 作为目标。

可以在"ip"或"ether"后面使用"multicast"及"broadcast"关键字。
当您想排除广播请求时，"no broadcast"就会非常有用。

查看 TCPdump 的主页以获得更详细的捕捉过滤器语法说明。
在 Wiki Wireshark website 上可以找到更多捕捉过滤器的例子。

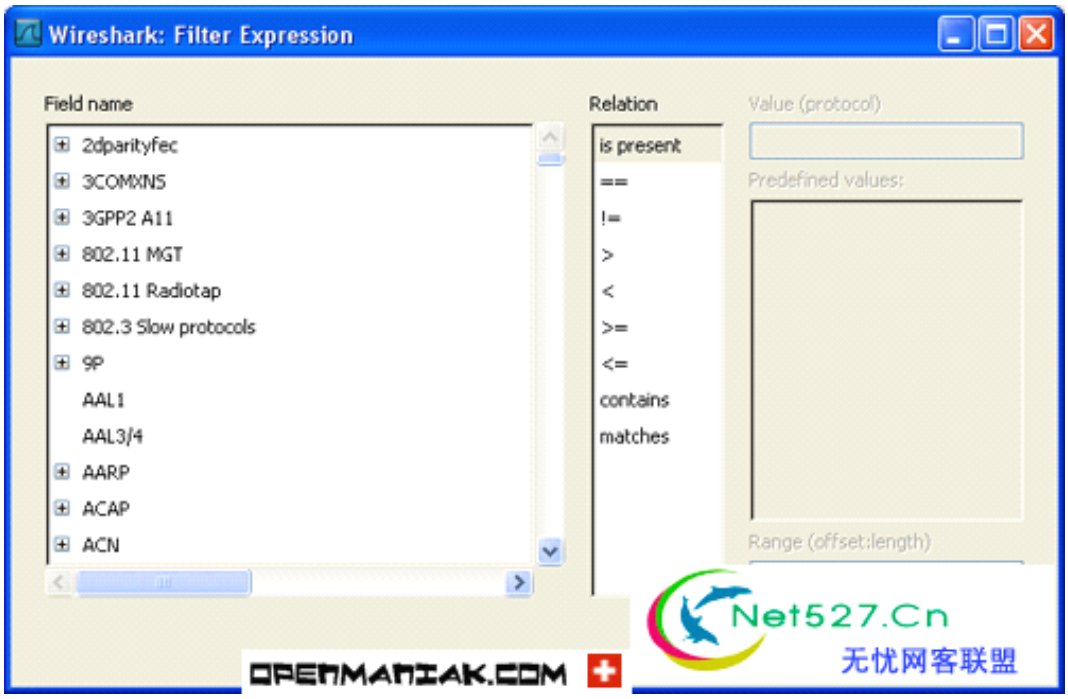
2. 显示过滤器：

通常经过捕捉过滤器过滤后的数据还是很复杂。此时您可以使用显示过滤器进行更加细致的查找。
它的功能比捕捉过滤器更为强大，而且在您想修改过滤器条件时，并不需要重新捕捉一次。

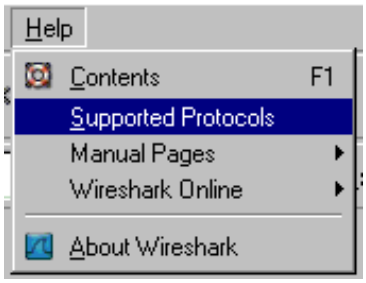
语法：	**Protocol**	.	**String 1**	.	**String 2**	**Comparison operator**	**Value**	**Logical Operations**	**Other expression_r**
例子：	ftp		passive		ip	==	10.2.3.4	xor	icmp.type

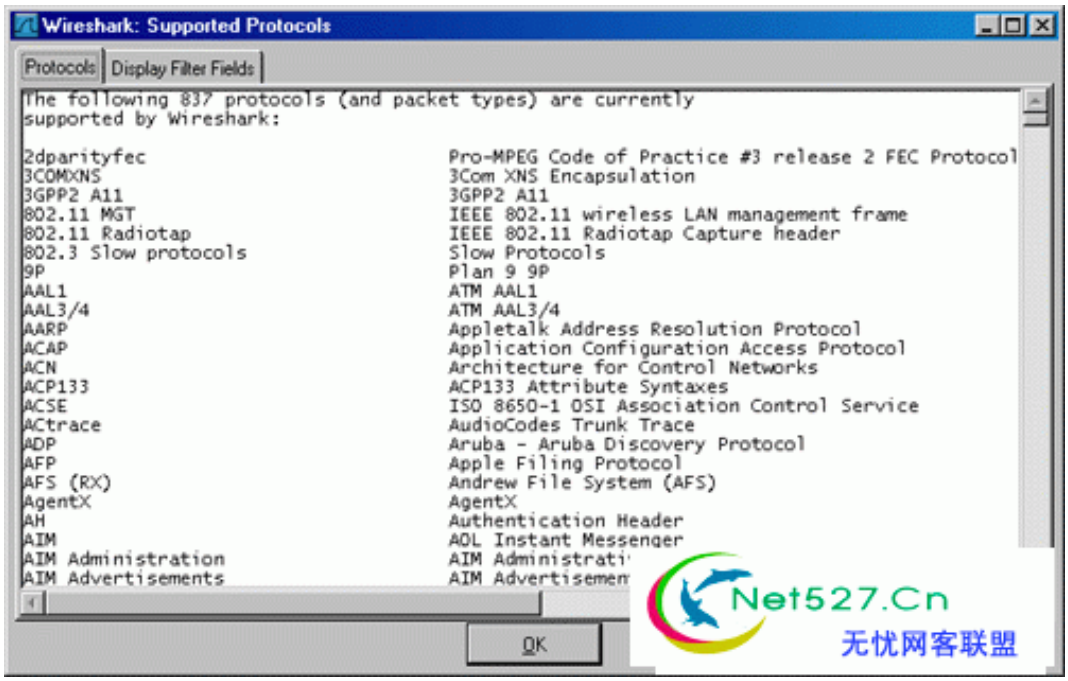
Protocol（协议）：

您可以使用大量位于 OSI 模型第 2 至 7 层的协议。点击"Expression..."按钮后，您可以看到它们。
比如：IP，TCP，DNS，SSH



您同样可以在如下所示位置找到所支持的协议：

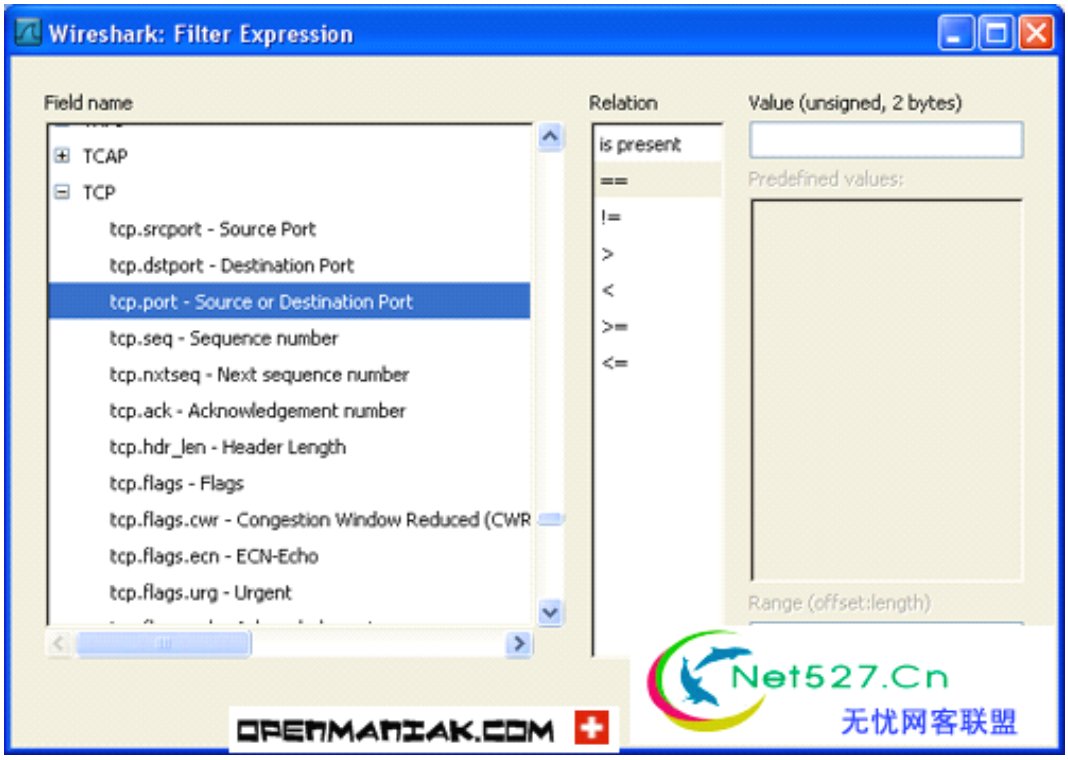




Wireshark 的网站提供了对各种 协议以及它们子类的说明。

String1, String2（可选项）:

协议的子类。
点击相关父类旁的"+"号，然后选择其子类。



Comparison operators （比较运算符）:

可以使用 6 种比较运算符：

英文写法：	C 语言写法：	含义：
eq	==	等于
ne	!=	不等于
gt	>	大于
lt	<	小于
ge	>=	大于等于
le	<=	小于等于

Logical expression_rs（逻辑运算符）:

英文写法：	C 语言写法：	含义：
and	&&	逻辑与
or	||	逻辑或

xor	^^	逻辑异或
not	!	逻辑非

被程序员们熟知的逻辑异或是一种排除性的或。当其被用在过滤器的两个条件之间时，只有当且仅当其中的一个条件满足时，这样的结果才会被显示在屏幕上。
让我们举个例子：
"tcp.dstport 80 xor tcp.dstport 1025"
只有当目的 TCP 端口为 80 或者来源于端口 1025（但又不能同时满足这两点）时，这样的封包才会被显示。

例子：

snmp || dns || icmp 显示 SNMP 或 DNS 或 ICMP 封包。

ip.addr == 10.1.1.1
显示来源或目的 IP 地址为 10.1.1.1 的封包。

ip.src != 10.1.2.3 or ip.dst != 10.4.5.6
显示来源不为 10.1.2.3 或者目的不为 10.4.5.6 的封包。
换句话说，显示的封包将会为：
来源 IP：除了 10.1.2.3 以外任意；目的 IP：任意
以及
来源 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
显示来源不为 10.1.2.3 并且目的 IP 不为 10.4.5.6 的封包。
换句话说，显示的封包将会为：
来源 IP：除了 10.1.2.3 以外任意；同时须满足，目的 IP：除了 10.4.5.6 以外任意

tcp.port == 25 显示来源或目的 TCP 端口号为 25 的封包。

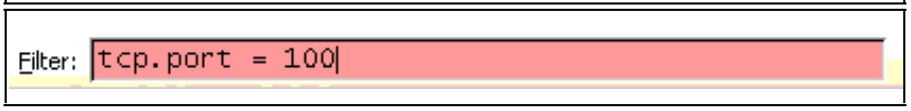
tcp.dstport == 25 显示目的 TCP 端口号为 25 的封包。

tcp.flags 显示包含 TCP 标志的封包。

tcp.flags.syn == 0x02 显示包含 TCP SYN 标志的封包。

如果过滤器的语法是正确的，表达式的背景呈绿色。如果呈红色，说明表达式有误。

表达式正确



表达式错误

(责任编辑：admin)