Wireshark 简介及说明

1.什么是 wireshark

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包，并为用户提供关于网络和上层协议的各种信息，并尝试显示包尽可能详细的情况。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。Wireshark 可能算得上是今天能使用的最好的开源网络分析软件。

Wireshark 的前身叫做 Ethereal（因为商标问题而改名），是开放源代码软件，我们可以免费从官方网站（http://wireshark.org）下载使用。Wireshark 支持多种操作系统，在 windows,UNIX,MAC 等系统都有相应的版本。通过此软件，我们可以抓取网络数据包，分析数据包的内容，一般常用在网络故障排除、软件封包检测。在使用上，Wireshark 提供的图形界面，非常直观，而且相当容易上手，以及支持丰富的过滤语言，可以灵活捕获我们需要的数据包，是一个非常有用的软件。

如果您不使用图形界面，也许会对 TShark 感兴趣。它是 Wireshark 的 CLI 版本。TShark 支持与 Wireshark 相同的功能。

下面是 Wireshark 一些应用的举例：

1.网络管理员用来解决网络问题

2.网络安全工程师用来检测安全隐患

3.开发人员用来测试协议执行情况

4.用来学习网络协议

除了上面提到的，Wireshark 还可以用在其它许多场合。

注意事项：

Wireshark 支持 UNIX 和 Windows 平台，对于 UNIX/Linux:Wireshark 依赖于 libpcap 库，它需要使用这个库的功能进行封包捕捉工作。 对于 Windows 平台:依赖于 winpcap 库。如果系统中没有安装 libpcap 或者 winpcap 库或者其他必要的组件，Wireshark 就无法工作，请自行安装。

使用 wireshark 的前提：

使用 wireshark 之前，请确保了解 OSI 七层网络协议，以及各层协议包首部的格式，否则看不懂捕获的信息。

wireshark 不能做的：

为了安全考虑，wireshark 只能查看封包，而不能修改封包的内容，或者发送封包。

Wireshark VS Fiddler:

Fiddler 是在 windows 上运行的程序，专门用来捕获 HTTP，HTTPS 的。

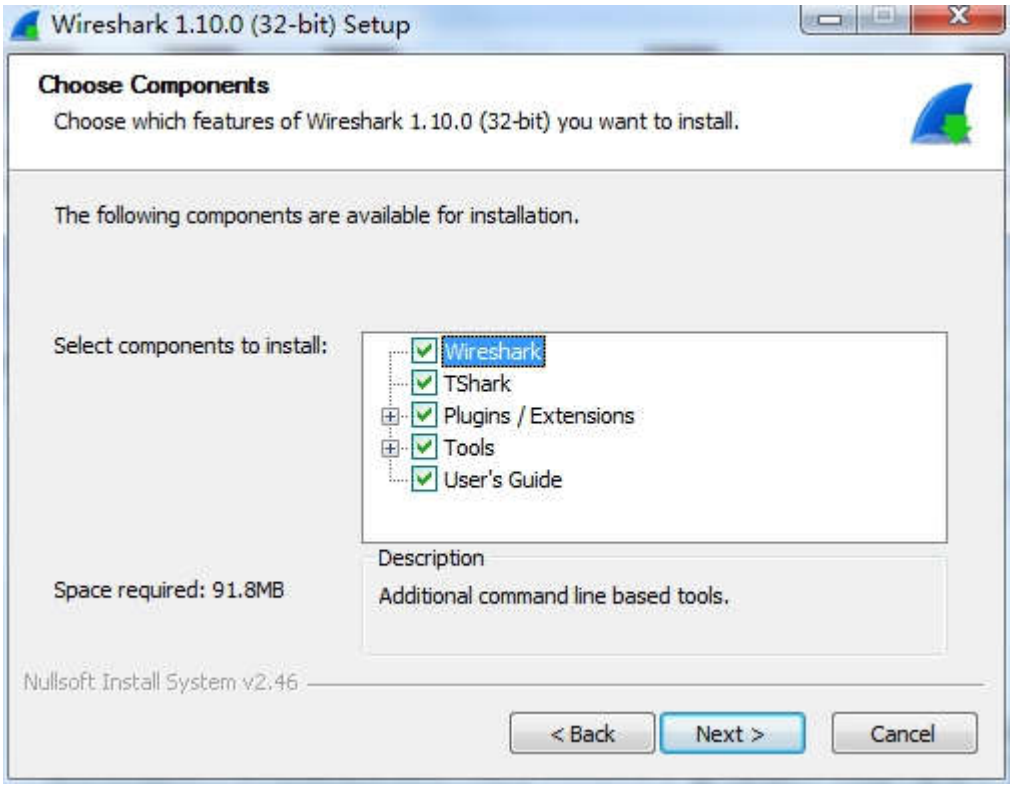
wireshark 能获取 HTTP，也能获取 HTTPS，但是不能解密 HTTPS，所以 wireshark 看不懂 HTTPS 中的内容

总结，如果是处理 HTTP,HTTPS 还是用 Fiddler，其他协议比如 TCP,UDP 就用 wireshark

2.安装 Wireshark

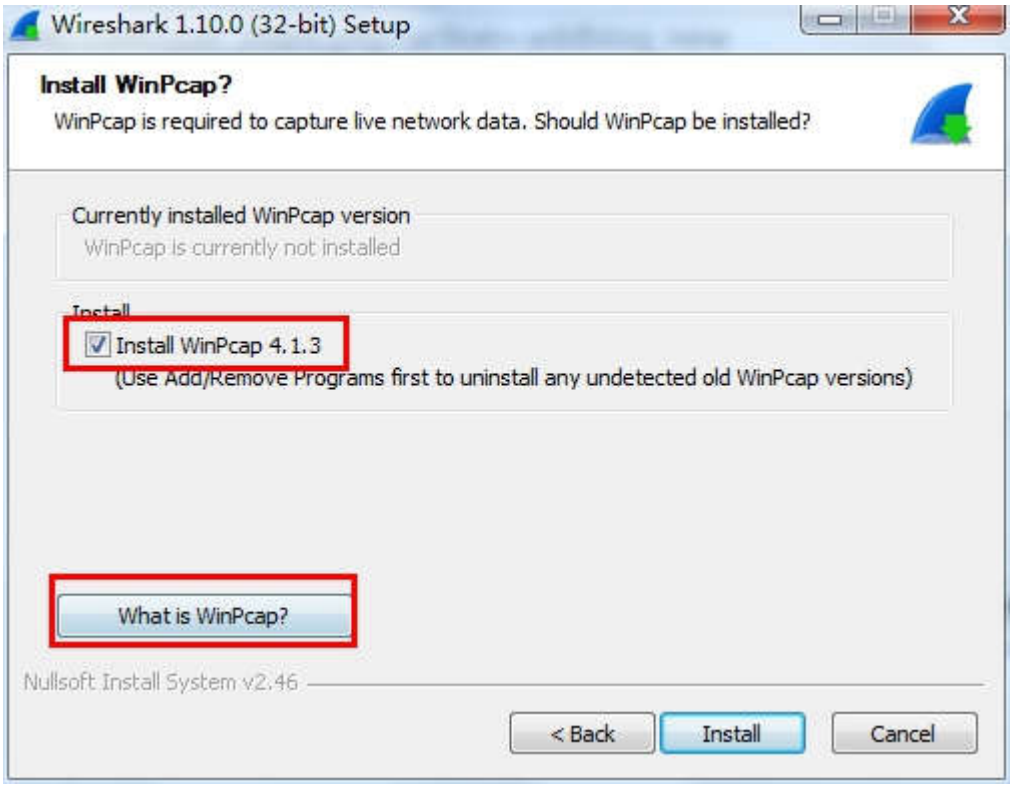
首先，我们到其官方网站（http://www.wireshark.org）下载相应操作系统版本的 wireshark 软件。这里以 windows 平台演示，安装的过程很简单，基本保持默认的设定即可。

注意：我只是截一些比较关键步骤的图。



需要注意的是，安装过程中会提示我们安装 WinPcap（Windows Packet Capture）,WinPcap 是 Windows 版本的 libpcap 库，Wireshark 使用这个库区抓取网络上的封包，它含有支持抓取网络封包的驱动程序，所以我们必须安装它。如果电脑上已经安装 WinPcap，也建议更新到最新的版本。

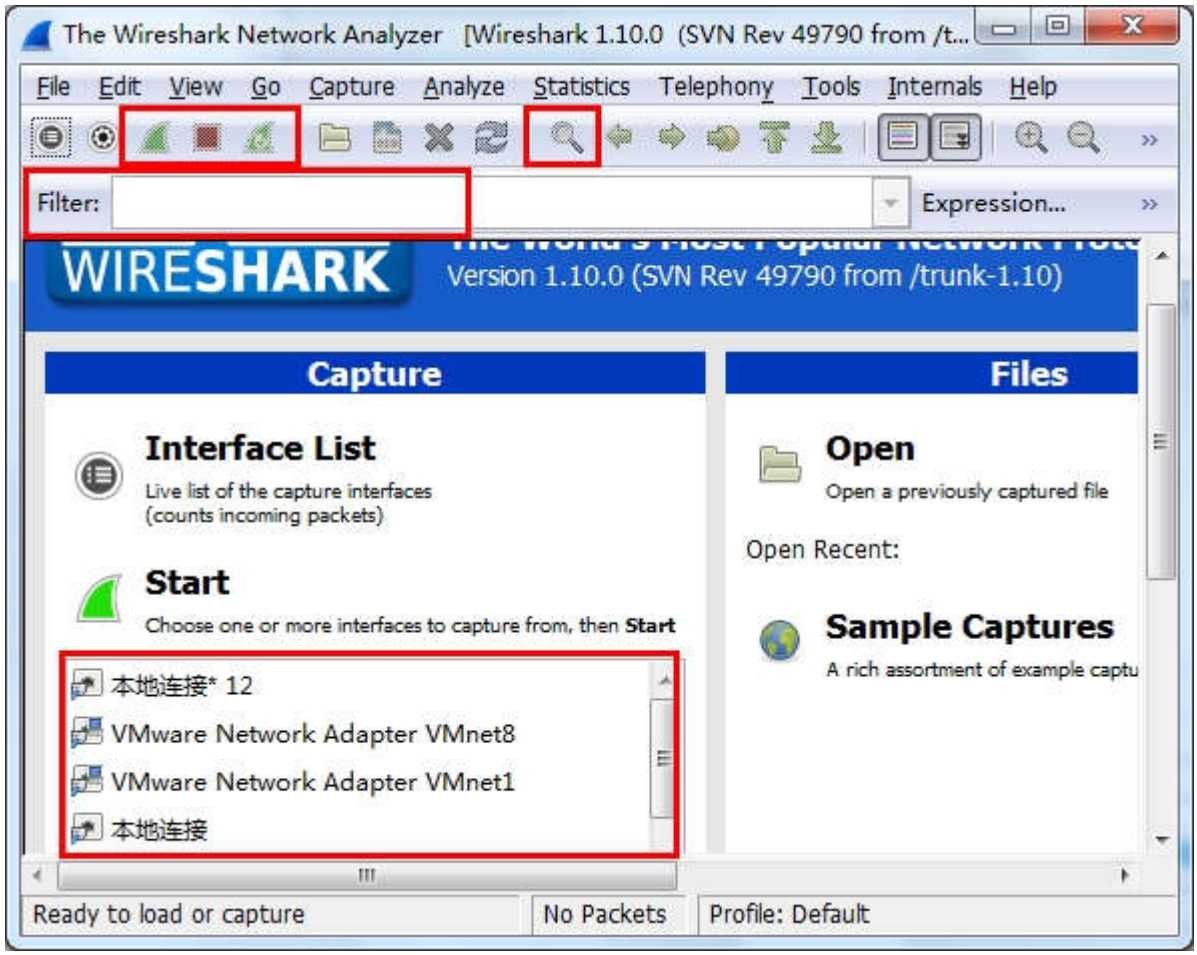
这里，请选择 Install WinPcap， 如果想了解更多的 WinPcap 信息，点击【What is WinPcap?】即可。 其他的地方都是一路 Next 即可



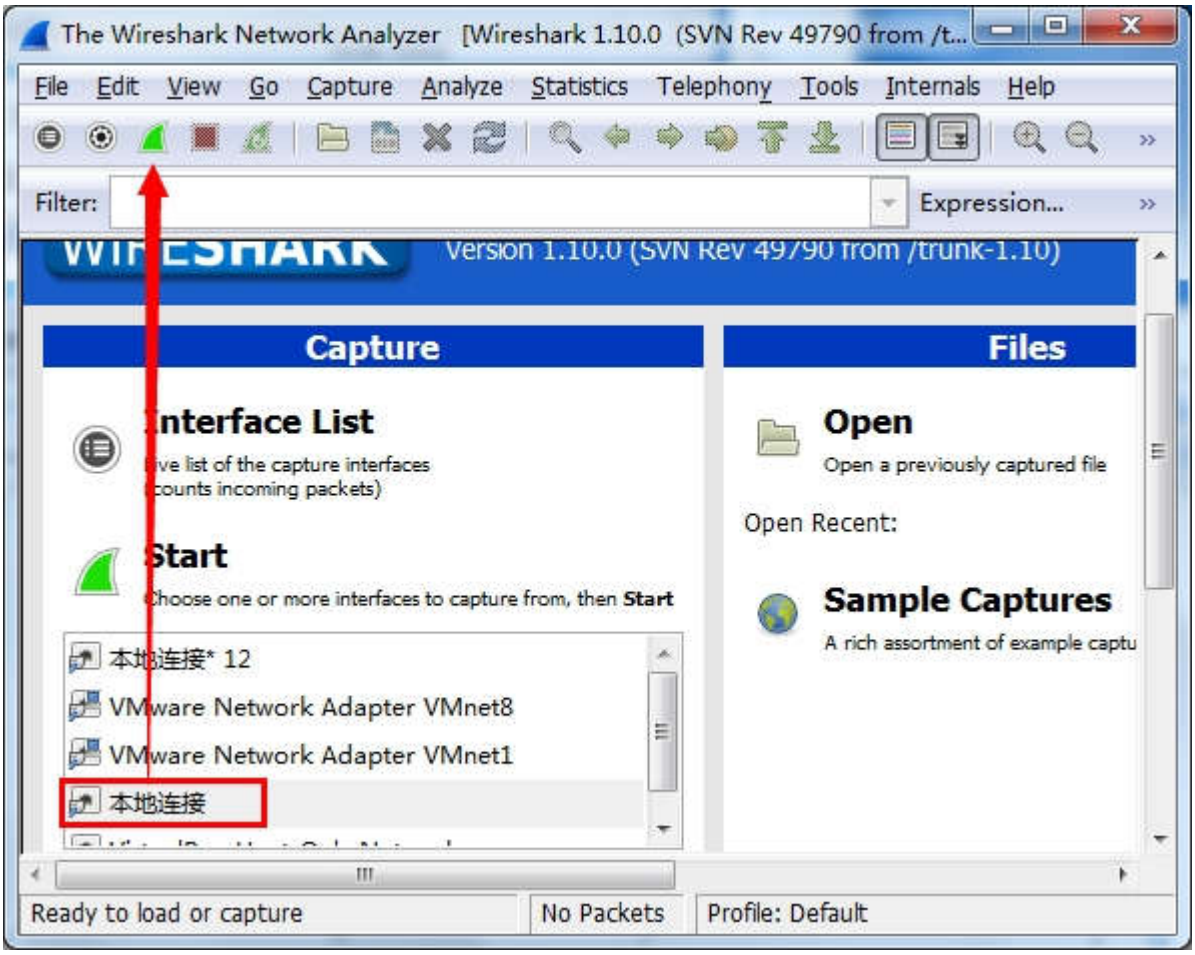
运行 Wireshark

Wireshark 开始页面，图中画框的地方，是我们最常用到的操作。

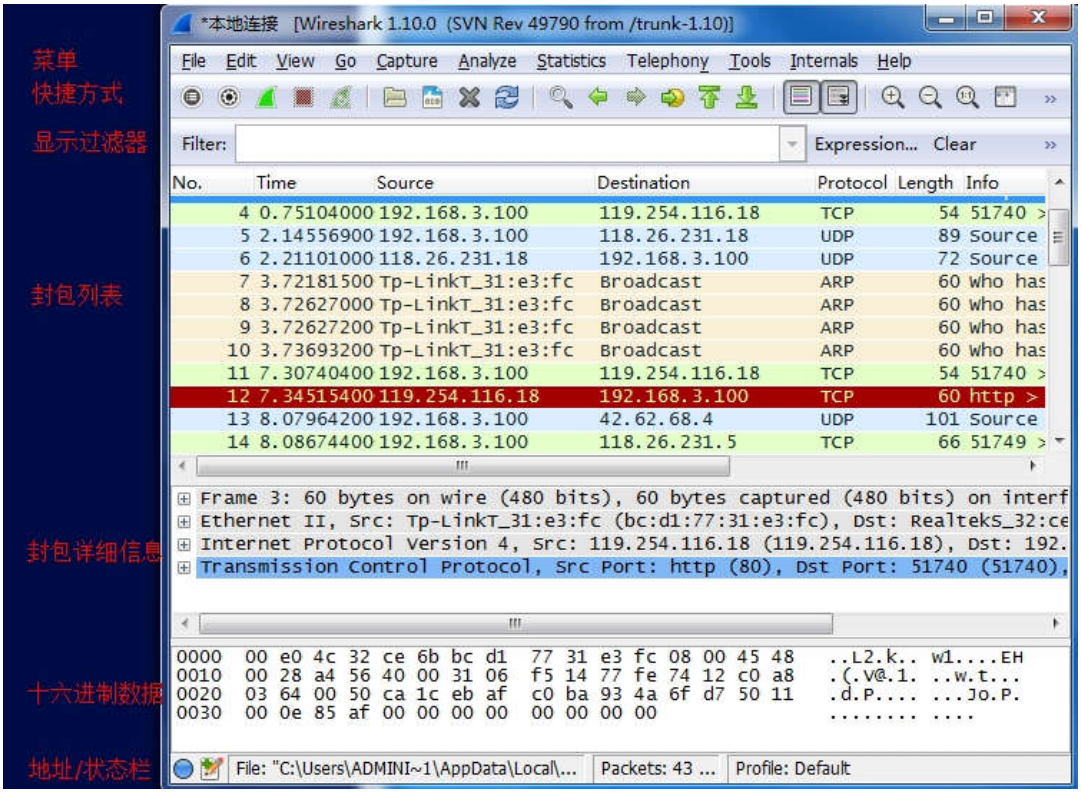
3.wireshark 使用说明



Wireshark 只能捕获机器某一块网卡的网络包，当你的机器有多块网卡的时候，需要指定捕获哪一块网卡的数据包。我们选择一块网卡之后，点击"Start"按钮，即可开始抓包。



下面我们看一下 wireshark 的窗口



我们分别介绍解释这些窗口的作用。

1、MENUS（菜单）

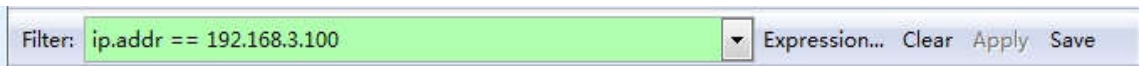
程序上方的 11 个菜单项用于对 Wireshark 进行配置：

- "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"(视图)　　设置 Wireshark 的视图。
- "Go"(转到)　　跳转到捕获的数据。
- "Capture"(捕获)　　设置捕获的网卡，捕获过滤器。
- "Analyze"(分析)　　设置分析选项。
- "Statistics"(统计)　　查看 wireshark 的统计信息
- "Telephony"(电话)
- "Tools"(工具)　　访问控制列表 ACL 及 LUA 相关
- "Internals"(内部信息)　解析表及支持的协议类型。
- "Help"(帮助)　　查看本地或者在线帮助文档。

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。

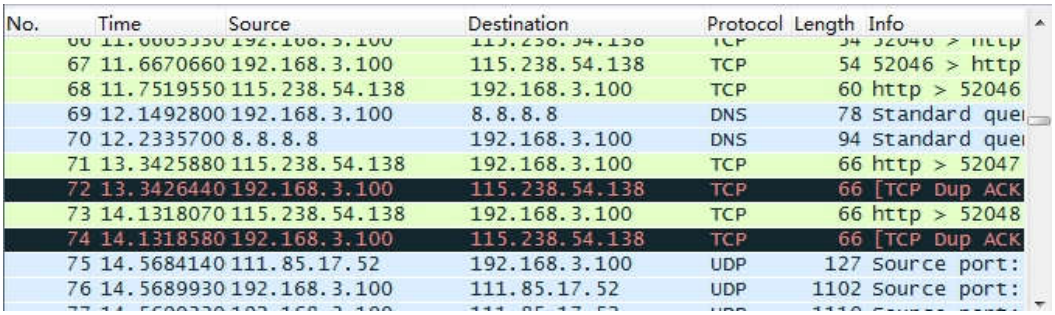
3、DISPLAY FILTER(显示过滤器)



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。如果过滤表达式正确，则方框中显示为淡绿色，表达式有错误，则为粉红色。

4、PACKET LIST PANE（封包列表）



No.	Time	Source	Destination	Protocol	Length	Info
67	11.6670660	192.168.3.100	115.238.54.138	TCP	54	52046 > http
68	11.7519550	115.238.54.138	192.168.3.100	TCP	60	http > 52046
69	12.1492800	192.168.3.100	8.8.8.8	DNS	78	Standard que
70	12.2335700	8.8.8.8	192.168.3.100	DNS	94	Standard que
71	13.3425880	115.238.54.138	192.168.3.100	TCP	66	http > 52047
72	13.3426440	192.168.3.100	115.238.54.138	TCP	66	[TCP Dup ACK
73	14.1318070	115.238.54.138	192.168.3.100	TCP	66	http > 52048
74	14.1318580	192.168.3.100	115.238.54.138	TCP	66	[TCP Dup ACK
75	14.5684140	111.85.17.52	192.168.3.100	UDP	127	Source port:
76	14.5689930	192.168.3.100	111.85.17.52	UDP	1102	Source port:

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。

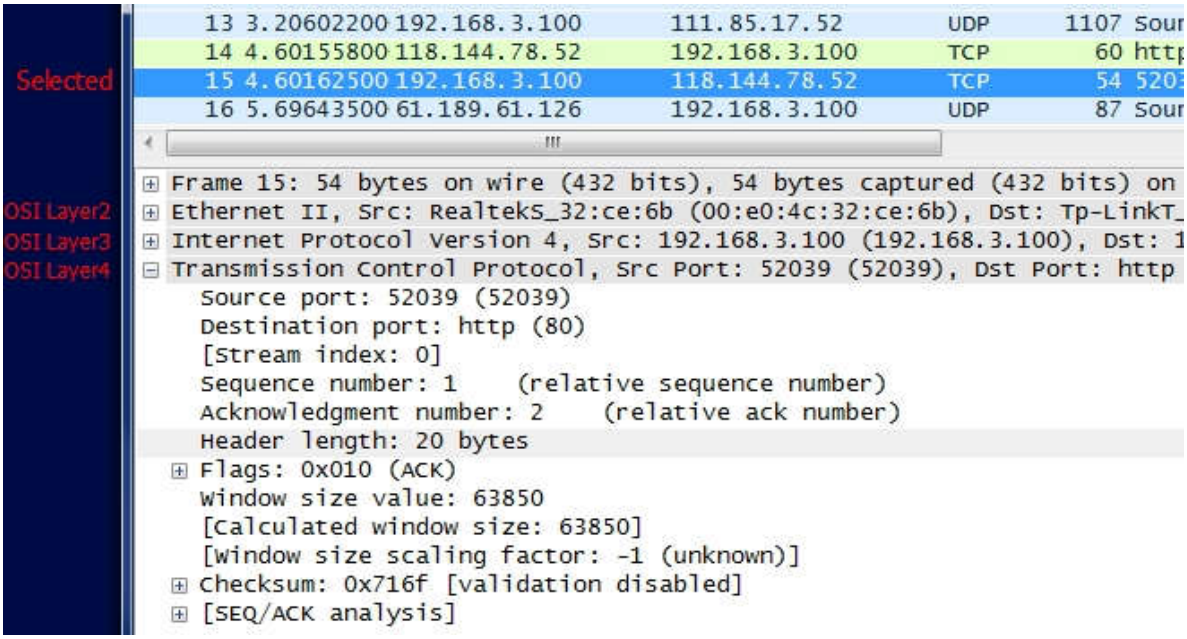
封包列表的面板中显示，编号，时间戳，源地址，目标地址，协议，长度，以及封包信息。你可以看到每一种协议都会以不同的颜色表示，你也可以修改这些颜色显示的规则：

View -> Coloring Rules

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。前面带" + "号的项，就可以展开。

各行信息分别为

Frame: 物理层的数据帧概况

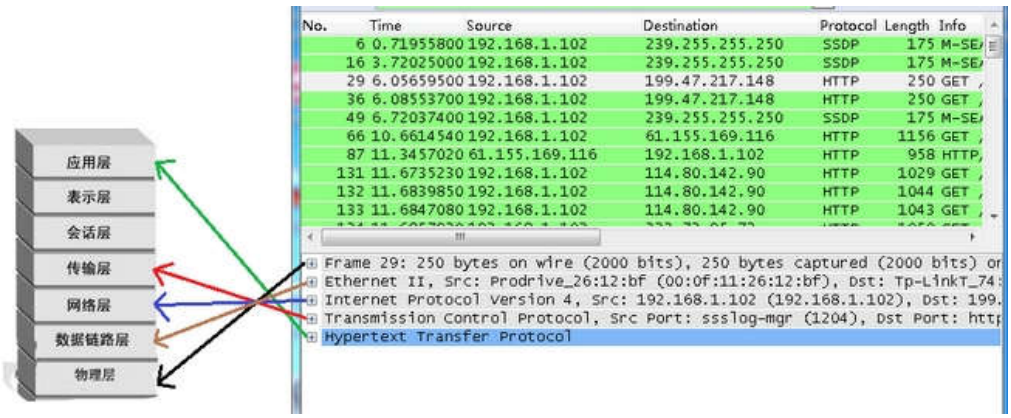
Ethernet II: 数据链路层以太网帧头部信息

Internet Protocol Version 4: 互联网层 IP 包头部信息

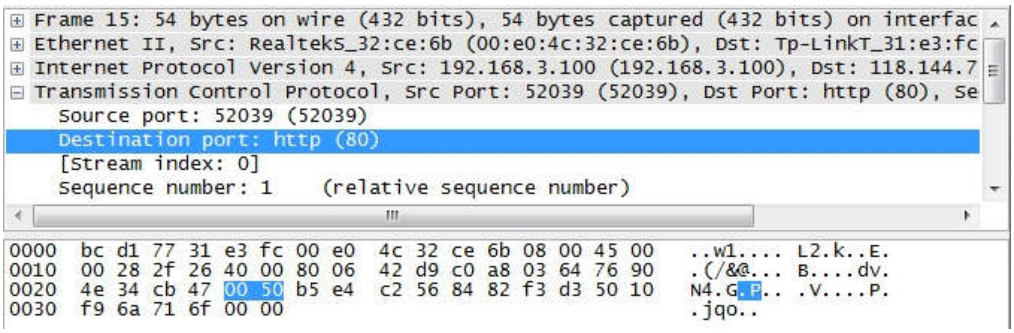
Transmission Control Protocol: 传输层 T 的数据段头部信息，此处是 TCP

Hypertext Transfer Protocol: 应用层的信息，此处是 HTTP 协议

wireshark 详细信息的层与对应 OSI 七层模型



6、DISSECTOR PANE（十六进制数据）



“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：

- "File:" 捕捉结果的保存位置。

- "Packets:" 已捕捉封包的数量。

- "Displayed:" 显示的封包数量。

- "Dropeed:" (经过显示过滤器过滤后仍然显示的封包)

Wireshark 之过滤器

使用 Wireshark 时最常见的问题，是当您使用默认设置时，瞬间就会得到

大量冗余信息，以至于很难找到自己需要的部分。

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。

捕获过滤器：用于决定捕获什么数据，需要在开始之前捕获时设置

显示过滤器：在捕获的结果中进行详细查找。它们可以在得到捕获结果后随意修改

那么我应该使用哪一种过滤器呢？

首先，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。

其次，两种过滤器使用的语法是完全不同的。

1.捕获过滤器

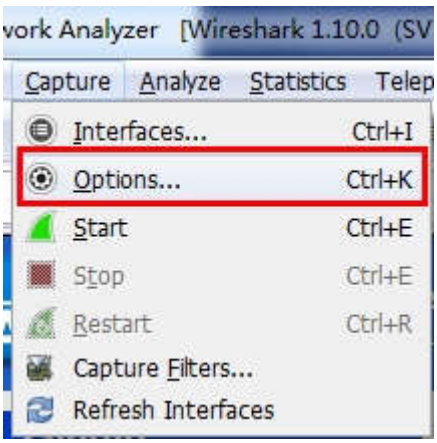
捕捉过滤器的语法与其它使用 Lipcap (Linux) 或者 Winpcap (Windows) 库开发的软件一样，比如著名的 TCPdump（如果您使用过 tcpdump，那么这个对于您来说，这都不是事儿）。捕捉过滤器必须在开始捕捉前设置完毕，这一点跟显示过滤器是不同的。

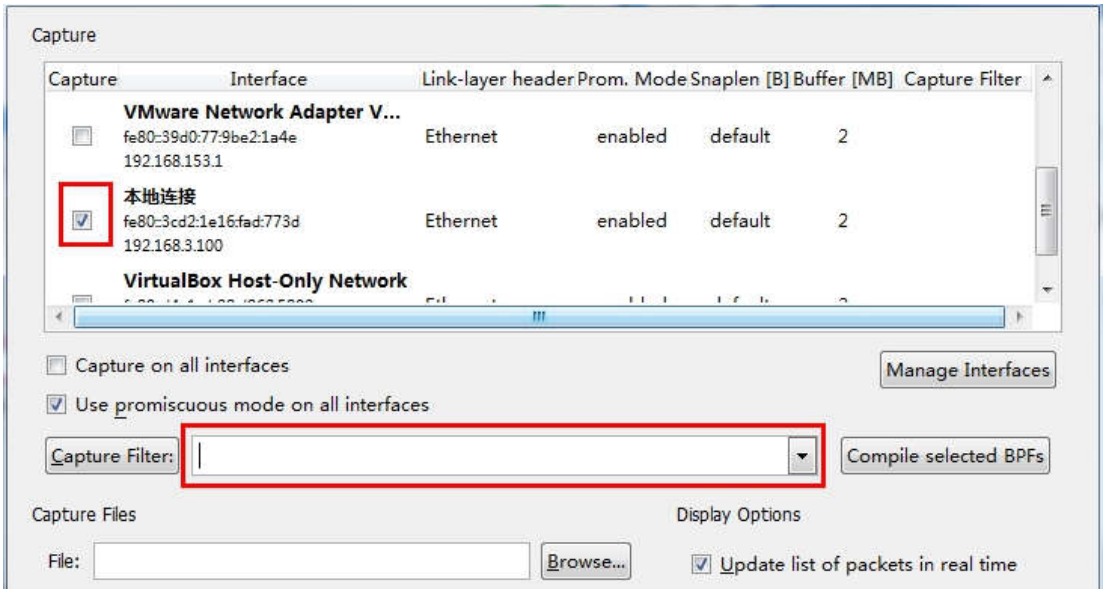
设置捕捉过滤器的步骤是：

- 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。





过滤表达式的语法(语法正确：浅绿色　语法错误：粉红色)

Syntax:	Protocol	Direction	Host(s)	Value	Logical Operations	Other expression
Example:	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：

```
// 显示目的 TCP 端口为 3128 的包
tcp dst port 3128


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

```
// 显示来源为 UDP 或 TCP， 并且端口号在 2000-2500 范围内的封包
src protrange 2000-2500


// 显示除了 icmp 以外的所有封包，icmp 通常被 ping 工具使用
not icmp     或者  ! icmp


// 显示来源 IP 地址为 10.7.2.12，但目的地址不是 10.200.0.0/16 的封包
src host 10.7.2.12 and (not dst net 10.200.0.0/16)
```

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。

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

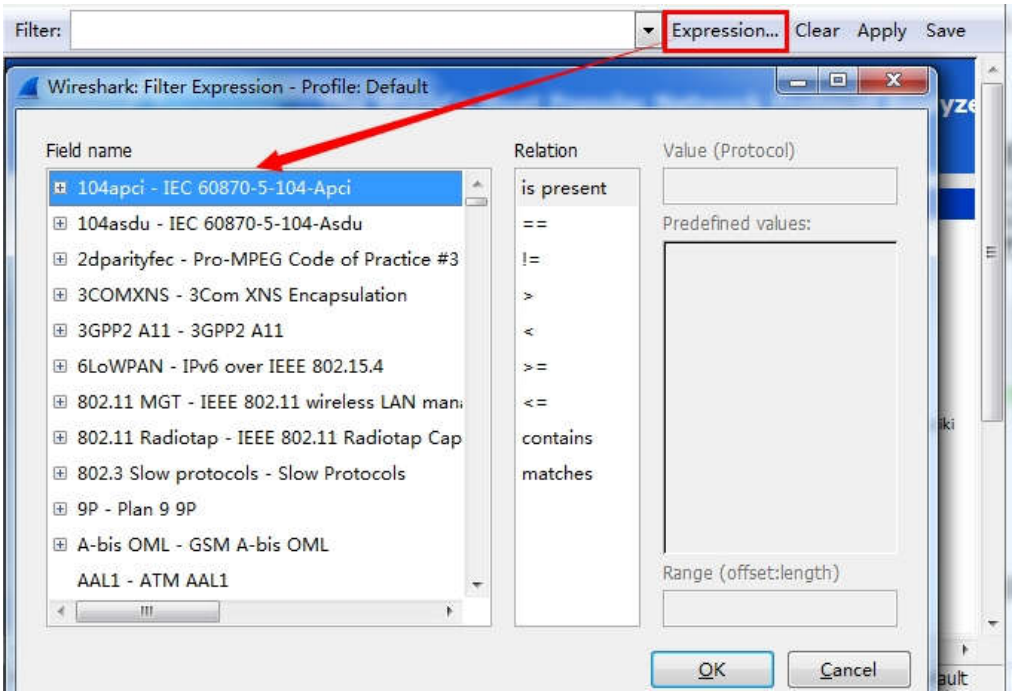
重新捕捉一次。

Syntax:	Protocol	.	String 1	.	String 2	Comparison operator	Value	Logical Operations	Other expression
Example:	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。

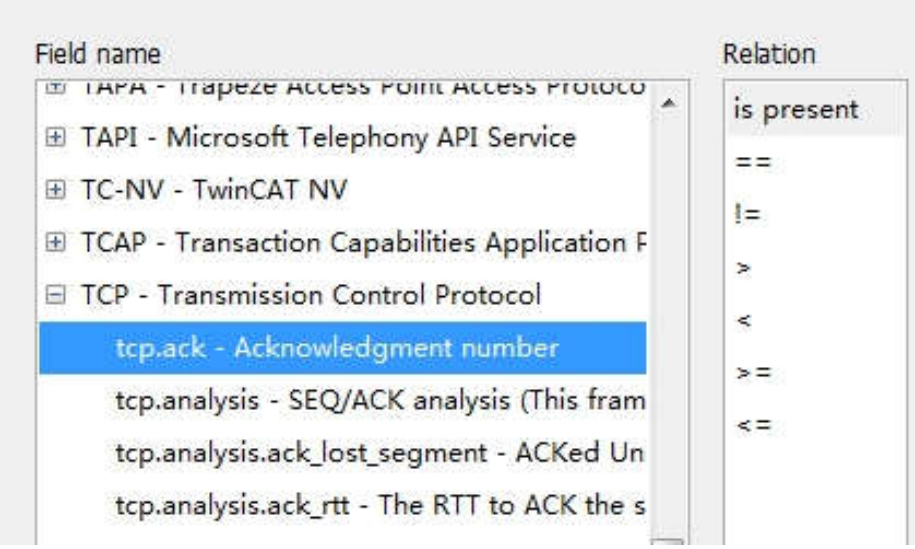
比如：IP，TCP，DNS，SSH



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 expressions（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：

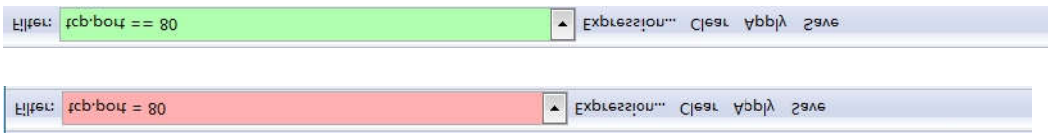
"tcp.dstport 80 xor tcp.dstport 1025"

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：

```
// 显示 SNMP 或 DNS 或 ICMP 封包
snmp || dns || icmp

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

// 显示来源不为 10.1.12.3 或者目的不为 10.4.5.6 的封包。
ip.src != 10.1.2.3 or ip.dst != 10.4.5.6

// 显示来源或目的 TCP 端口为 25 的封包
tcp.port == 25

// 显示目的 TCP 端口为 25 的封包
tcp.dstport == 25

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

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。



最后补充下，输入过滤表达式之后，点击"Apply"应用生效。如果想重新过滤的话，不能在过滤结果中再打过滤条件，记得"Clear"掉过滤条件后再打。