[wireshark的使用教程--用实践的方式帮助我们理解TCP/IP中的各个协议是如何工作的](http://blog.csdn.net/21aspnet/article/details/6667431)

分类： [TCP/IP协议](http://blog.csdn.net/21aspnet/article/category/837911)2011-08-07 19:37 7220人阅读 [评论](http://blog.csdn.net/21aspnet/article/details/6667431#comments)(3) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/21aspnet/article/details/6667431#report)

[工作](http://www.csdn.net/tag/%e5%b7%a5%e4%bd%9c)[tcp](http://www.csdn.net/tag/tcp)[dst](http://www.csdn.net/tag/dst)[工具](http://www.csdn.net/tag/%e5%b7%a5%e5%85%b7)[stream](http://www.csdn.net/tag/stream)[statistics](http://www.csdn.net/tag/statistics)

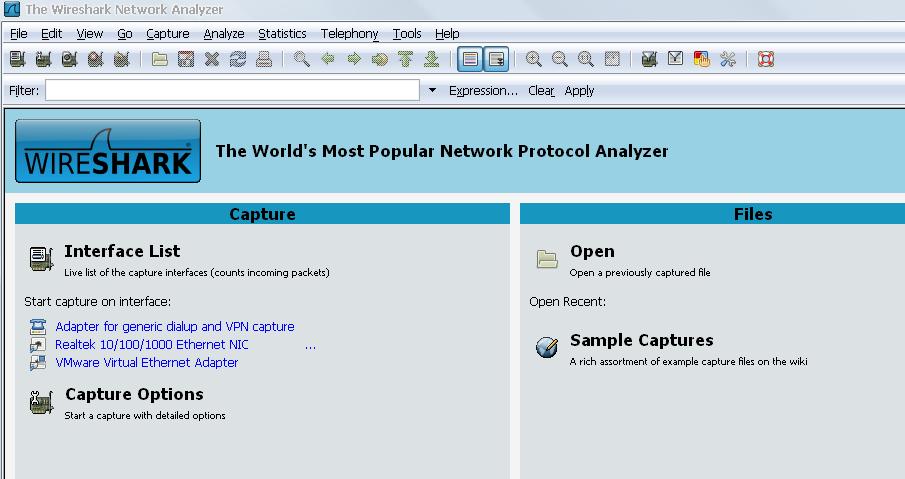
**wireshark的使用教程**

**--用实践的方式帮助我们理解TCP/IP中的各个协议是如何工作的**

wireshark是一款抓包软件，比较易用，在平常可以利用它抓包，分析协议或者监控网络，是一个比较好的工具，因为最近在研究这个，所以就写一下教程，方便大家学习。

       这里先说Wireshark的启动界面和抓包界面

       启动界面：

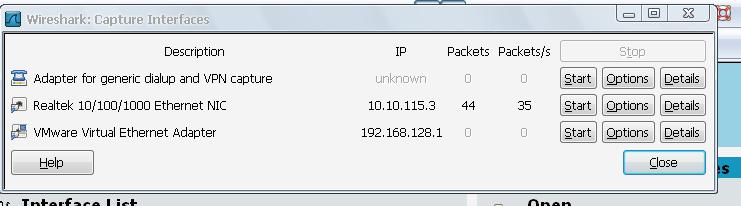


抓包界面的启动是

按file下的按钮

http://hi.csdn.net/attachment/201108/7/0_1312716699xe00.gif

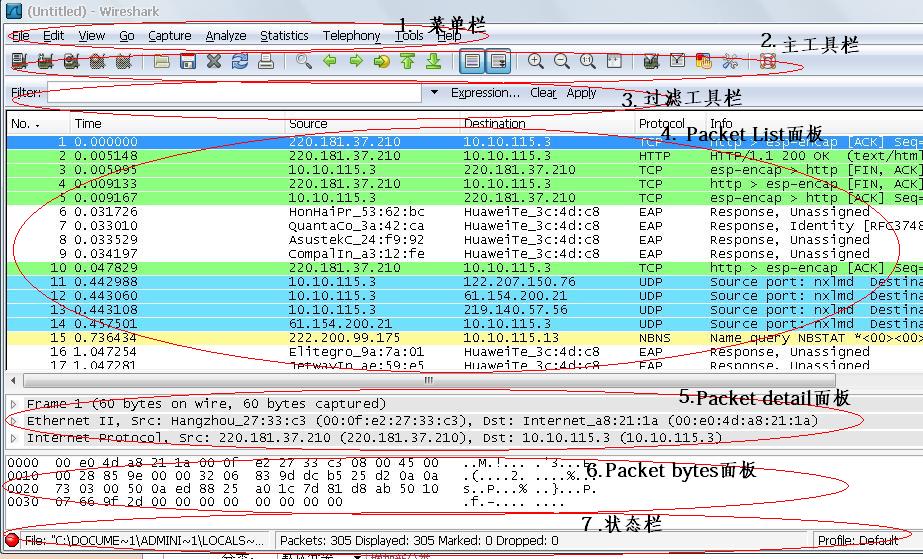
之后会出现



这个是网卡的显示，因为我有虚拟机所以会显示虚拟网卡，我们现在抓的是真实网卡上的包所以在以太网卡右边点击start 开始抓包

http://hi.csdn.net/attachment/201108/7/0_1312716753JZ7Z.gif

这个就是抓包的界面了（也是主界面）



Wireshark主窗口由如下部分组成：

1. 菜单——用于开始操作。
2. 主工具栏——提供快速访问菜单中经常用到的项目的功能。
3. Fiter toolbar/过滤工具栏——提供处理当前显示过滤得方法。
4. Packet List面板——显示打开文件的每个包的摘要。点击面板中的单独条目，包的其他情况将会显示在另外两个面板中。
5. Packet detail面板——显示您在Packet list面板中选择的包的更多详情。
6. Packet bytes面板——显示您在Packet list面板选择的包的数据，以及在Packet details面板高亮显示的字段。
7. 状态栏——显示当前程序状态以及捕捉数据的更多详情。

**1.菜单栏**

主菜单包括以下几个项目:

File ——包括打开、合并捕捉文件，save/保存,Print/打印,Export/导出捕捉文件的全部或部分。以及退出Wireshark项.

Edit ——包括如下项目：查找包，时间参考，标记一个多个包，设置预设参数。（剪切，拷贝，粘贴不能立即执行。）

View ——控制捕捉数据的显示方式，包括颜色，字体缩放，将包显示在分离的窗口，展开或收缩详情面版的地树状节点

GO ——包含到指定包的功能。

Analyze ——包含处理显示过滤，允许或禁止分析协议，配置用户指定解码和追踪TCP流等功能。

Statistics ——包括的菜单项用户显示多个统计窗口，包括关于捕捉包的摘要，协议层次统计等等。

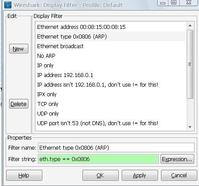
Help ——包含一些辅助用户的参考内容。如访问一些基本的帮助文件，支持的协议列表，用户手册。在线访问一些网站，“关于”

**2.工具栏（略）**

**3.过滤工具栏**

http://hi.csdn.net/attachment/201108/7/0_1312716809n1CV.gif

http://hi.csdn.net/attachment/201108/7/0_1312716825VGbz.gif会弹出对话框



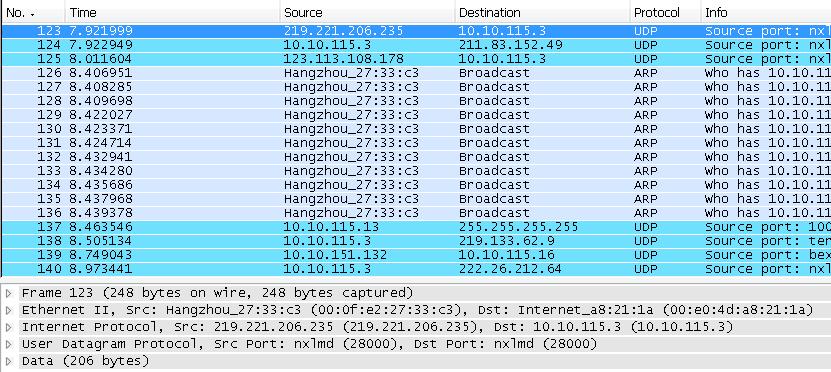
这个和在工具栏上输入协议来查找包的结果是一样的，只是它方便点

在工具栏上输http://hi.csdn.net/attachment/201108/7/0_131271691011sM.gif

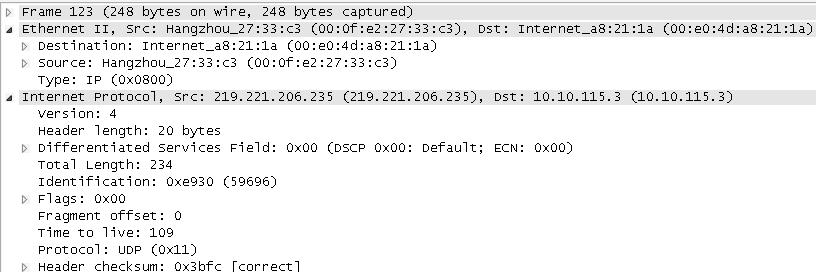
点击 在此区域输入或修改显示的过滤字符，在输入过程中会进行语法检查。如果您输入的格式不正确，或者未输入完成，则背景显示为红色。直到您输入合法的表达式，背景会变为绿色。你可以点击下拉列表选择您先前键入的过滤字符。列表会一直保留，即使您重新启动程序。

 注意：

* 1. 做完修改之后，记得点击右边的Apply(应用)按钮，或者回车，以使过滤生效。
  2. 输入框的内容同时也是当前过滤器的内容（当前过滤器的内容会反映在输入框）

**4.封包列表** 

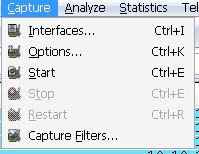
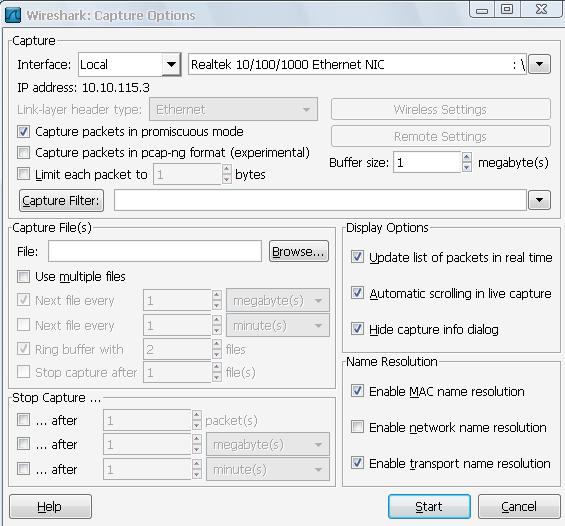
|  |
| --- |
| http://hi.csdn.net/attachment/201108/7/0_1312716952Ofx3.gif |

* 1. 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
  2. **5.封包详细信息**
  3.   
       
       
       
     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。
  4. ps：wireshark会用不难，难的是会看懂这些包  
     （如果对这些包头信息不了解的可以自己去查查资料）
  5. **6. 16进制数据**

|  |
| --- |
| http://hi.csdn.net/attachment/201108/7/0_13127169886s3k.gif |

* 1. “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。
  2. 在这里我们还要学会在大量的信息中**过滤**和**分析**我们需要的信息
  3. 过滤：
  4. 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。 |

* 1. 两种过滤器的目的是不同的。
  2. 1.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。
  3. 2.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。
  4. **1.捕捉过滤器**
  5. 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](http://openmaniak.com/tcpdump.php)。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 |

* 1. http://openmaniak.com/image/droite.gif **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。  
       
     http://openmaniak.com/image/droite.gif **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。  
     http://openmaniak.com/image/droite.gif **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。  
       
     http://openmaniak.com/image/droite.gif **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。
  2. 例子：

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

* 1. 显示目的TCP端口为3128的封包。

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

* 1. 显示来源IP地址为10.1.1.1的封包。

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

* 1. 显示目的或来源IP地址为10.1.2.3的封包。

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

* 1. 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** |

* 1. 显示除了icmp以外的所有封包。（icmp通常被ping工具使用）

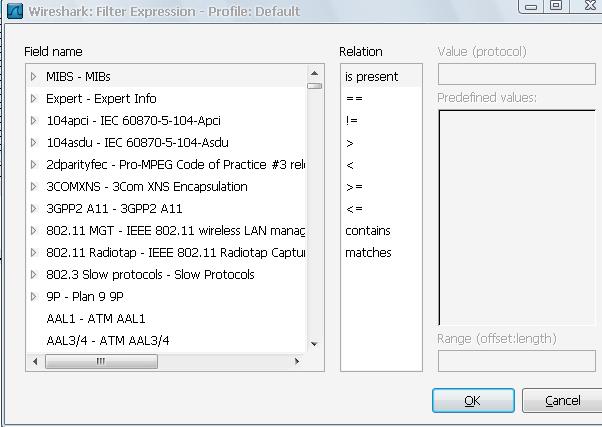
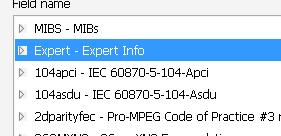
|  |
| --- |
| **src host 10.7.2.12 and not dst net 10.200.0.0/16** |

* 1. 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** |

* 1. 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。
  2. 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。
  3. **http://openmaniak.com/image/carre.gif 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 |

* 1. http://openmaniak.com/image/droite.gif **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  
       
       
       
       
       
       
       
     http://openmaniak.com/image/droite.gif **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。  
       
       
       
       
     http://openmaniak.com/image/droite.gif **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 |

* 1. http://openmaniak.com/image/droite.gif **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 |

* 1. 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。
  2. 例子：

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

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

* 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** |

* 1. 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** |

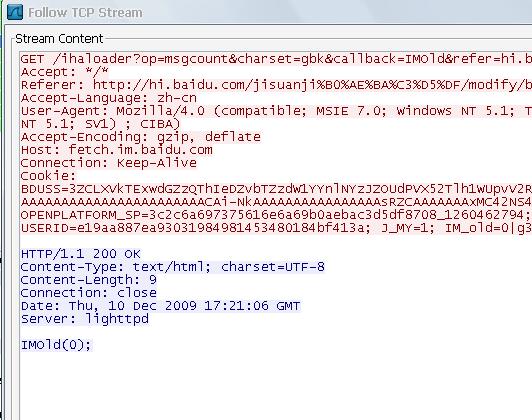
* 1. 显示来源不为10.1.2.3并且目的IP不为10.4.5.6的封包。  
     换句话说，显示的封包将会为：  
     来源IP：除了10.1.2.3以外任意；同时须满足，目的IP：除了10.4.5.6以外任意

|  |  |
| --- | --- |
| **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。 |

* 1. 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。
  2. **分析：**
  3. **"Follow TCP Stream"**
  4. 如果你处理TCP协议，想要查看Tcp流中的应用层数据，"Following TCP streams"功能将会很有用。如果你项查看telnet流中的密码，或者你想尝试弄明白一个数据流。或者你仅仅只需要一个显示过滤来显示某个TCP流的包。这些都可以通过Wireshark的"Following TCP streams"功能来实现。
  5. 在包列表中选择一个你感兴趣的TCP包，然后选择Wireshark工具栏菜单的"Following TCP Streams"选项(或者使用包列表鼠标右键的上下文菜单)。然后，Wireshark就会创建合适的显示过滤器，并弹出一个对话框显示TCP流的所有数据。
  6. **"Follow TCP Stream"对话框**
  7. 
  8. 流的内容出现的顺序同他们在网络中出现的顺序一致。从A到B的通信标记为红色，从B到A的通信标记为蓝色。当然，如果你喜欢的话你可以从"Edit/Preferences"菜单项的"Colores"修改颜色。
  9. 非打印字符将会被显示为圆点。XXX - What about line wrapping (maximum line length) and CRNL conversions?
  10. 在捕捉过程中，TCP流不能实时更新。想得到最近的内容需要重新打开对话框。
  11. 你可以在此对话框执行如下操作：
      + 1. **Save As**以当前选择格式保存流数据。
        2. **Print**以当前选择格式打印流数据。
        3. **Direction** 选择流的显示方向("Entire conversation", "data from A to B only" or "data from B to A only").
        4. **Filter out this stream** 应用一个显示过滤，在显示中排除当前选择的TCP流。
        5. **Close** 关闭当前对话框。移除对当前显示过滤的影响。

你可以用以下格式浏览流数据。

* + - 1. **AsCII**。在此视图下你可以以ASCII凡是查看数据。当然最适合基于ASCII的协议用，例如HTTP.
      2. **EBCDIC**。For the big-iron freaks out there.（不知道这句是什么意思， EBCDIC 是IBM公司的字符二进制编码标准。）
      3. **HEX Dump**. 允许你查看所有数据，可能会占用大量屏幕空间。适合显示二进制协议。
      4. **C Arrays.** 允许你将流数据导入你自己的C语言程序。
      5. **RAW**。 允许你载入原始数据到其他应用程序做进一步分析。显示类似与ASCII设置。但“save As”将会保存为二进制文件