i春秋实验课时1--认识Wireshark的界面

原创

NillLazy 于 2016-12-18 20:16:38 发布
分类专栏： 网络数据包分析从入门到精通


网络数据包分析从入门到精通 专栏收录该内容
0 篇文章 0 订阅
订阅专栏

实验环境

实验环境
操作机： Windows XP

实验工具：

Tools	Path
Wireshark	桌面

实验文件：
Lab1-1.pcapng

本课程我们将认识Wireshark的界面。

实验步骤

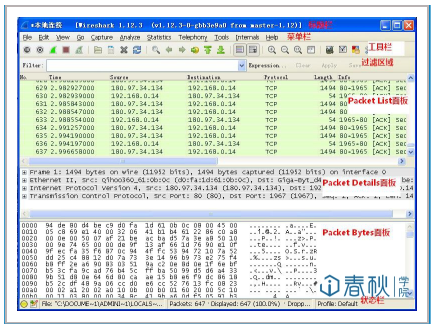
下载实验文件

请访问 http://file.ichunqiu.com/45668f52 下载**实验文件**。

> **小i提示：**
>
> 在本次实验中,请注意实验工具、实验文件存放路径,不同的文件路径可能会出现不一样的实验结果。
> 在实验环境中无法连接互联网，请使用您本地的网络环境。

Wireshark的主窗口界面

我们在正式利用 Wireshark 进行数据包分析之前，应当首先了解一下这款软件主 窗口界面中每个部分的功能。Wireshark主窗口界面如下图所示（实验文件Lab1- 1.pcapng）：



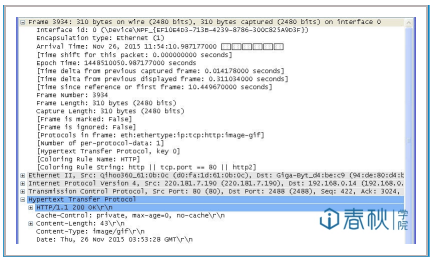
上图中每部分的含义如下：

标题栏：用于显示所分析的抓包文件的名称、捕获的设备名称以及Wireshark的 版本号。
菜单栏：Wireshark的标准菜单栏。
工具栏：常用功能的快捷图标按钮。
筛选区域：我们在实际的数据包分析中，可能在很短的时间内就能够捕获到成 千上万的数据包信息。这个时候就需要使用这里的 筛选器 ，加上一定的条件，筛 选掉我们并不关心的数据包，从而更好地进行分析。我们在接下来的实际分析课程 中，会多次使用筛选功能。
Packet List面板：显示每个数据帧的摘要。需要强调的是，其实这里所捕获 的其实就是 数据帧 ，但是出于表达的习惯，本系列的课程中的大部分时候，我会 以“数据包”的叫法来代替“数据帧”以及“分段”。这里采用表格的形式列出了 当前捕获文件中的所有数据包，其中包括了数据包 序号 、数据包捕获的 相对时 间 、数据包的 源地址 和 目标地址 、数据包的 协议 以及在数据包中找到的 概 况信息 等。
Packet Details面板：分析数据包的详细信息。这个面板分层次地显示了一个 数据包中的内容，并且可以通过展开或是收缩来显示这个数据包中所捕获的全部内 容。在我们的课程中， Packet List 面板以及 Packet Details 面板是我们重点 关注的对象。
Packet Bytes面板：以十六进制和ASCII码的形式显示数据包的内容。这里显 示了一个数据包未经处理的 原始 样子，也就是在链路上传播时的样子。
状态栏：包含有专家信息、注释、包的数量和Profile。

认识数据包

一旦我们开始利用Wireshark进行数据包的 捕获 ，那么所获得的二进制数据就会 依照不同的协议的结构进行规范，并且显示在 Packet Details 面板中。这里我 给大家简单介绍一下识别数据包的方法。

在Wireshark中，关于数据包的叫法主要有3个术语，也就是帧、包、段。数据帧 的起始点和目的点都是 数据链路层 ；数据包的起始和目的地是网络层；段通常是 指起始点和目的地都是传输层的信息单元。这里我们通过以下捕获的数据包来分析 一下这三种术语（以下展示的是通过筛选器显示 HTTP协议 ，并查看3934号数据 包的情况）：



这里面主要展示了以下五种协议的信息：

Frame：物理层数据帧的情况。
Ethernet II：数据链路层以太网帧头部的信息。
Internet Protocol Version 4：网络层IP包的头部信息。
Transmission Control Protocol：传输层的数据段头部信息，这里显示的是

TCP协议。

Hypertext Transfer Protocol：应用层信息，这里显示的是HTTP协议。 下面依据上述所捕获到的数据包的具体情况，对上述前四层的内容进行分析（HTTP 协议在以后的课程中会专门用一次课来讲解）：

物理层数据帧的情况

```
Frame 3934: 310 bytes on wire (2480 bits), 310 bytes captured (2480

bits) on interface 0
# 3934号帧，线路上有310个字节，实际捕获310个字节
Interface id: 0 (\Device\NPF_{EF10E4D3-713B-4239-8786-300C825A9D3F})
# 接口id
Encapsulation type: Ethernet (1)
# 封装类型
Arrival Time: Nov 26, 2015 11:54:10.987177000 中国标准时间
# 捕获日期和时间
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1448510050.987177000 seconds
[Time delta from previous captured frame: 0.014178000 seconds]
# 当前数据包与前一个数据包的时间间隔
[Time delta from previous displayed frame: 0.311034000 seconds]
[Time since reference or first frame: 10.449670000 seconds]
# 当前数据包与第一个数据包的时间间隔
Frame Number: 3934
# 帧序号
Frame Length: 310 bytes (2480 bits)
# 帧长度
Capture Length: 310 bytes (2480 bits)
# 捕获长度
[Frame is marked: False]
# 此帧是否做了标记：否
[Frame is ignored: False]
# 此帧是否被忽略：否
[Protocols in frame: eth:ethertype:ip:tcp:http:image-gif]
# 帧内封装的协议层次结构
[Number of per-protocol-data: 1]
[Hypertext Transfer Protocol, key 0]
[Coloring Rule Name: HTTP]
# 着色标记的协议名称
[Coloring Rule String: http || tcp.port == 80 || http2]
着色规则显示的字符串
```

数据链路层以太网帧头部的信息

```
Ethernet II, Src: Qihoo360_61:0b:0c (d0:fa:1d:61:0b:0c), Dst: Giga-

Byt_d4:be:c9 (94:de:80:d4:be:c9)
Destination: Giga-Byt_d4:be:c9 (94:de:80:d4:be:c9)
# 目标MAC地址
Source: Qihoo360_61:0b:0c (d0:fa:1d:61:0b:0c)
# 源MAC地址
Type: IP (0x0800)
# 需要说明的是，为什么上述的源MAC地址以及目标MAC地址的开头明明

是“d0:fa:1d”以及“94:de:80”，
# 但是Wireshark上显示出来的却

是“Qihoo360”以及“Giga-Byt”#呢？这是因为MAC地址的前3
# 个字节表示厂商。而“d0:fa:1d”以及“94:de:80”被分# 配给了奇虎以及技

嘉公司。这是全球统一标# 准，所以Wireshark干脆显示出厂商名称了
```

网络层IP包的头部信息

```
Internet Protocol Version 4, Src: 220.181.7.190 (220.181.7.190), Dst:


192.168.0.14 (192.168.0.14)
Version: 4
# 互联网协议IPv4
Header Length: 20 bytes
# IP包头部的长度
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00:

Not-ECT (Not ECN-Capable Transport))
# 差分服务字段
Total Length: 296
# IP包的总长度
Identification: 0x2863 (10339)
# 标志字段
Flags: 0x02 (Don't Fragment)
# 标记字段
Fragment offset: 0
# 分片偏移
Time to live: 49
# 生存期
Protocol: TCP (6)
# 当前数据包所封装的上层协议为TCP协议
Header checksum: 0x7b43 [validation disabled]
# 头部数据的校验和
Source: 220.181.7.190 (220.181.7.190)
# 源IP地址
Destination: 192.168.0.14 (192.168.0.14)
# 目标IP地址
```

传输层TCP数据段的头部信息

```
Transmission Control Protocol, Src Port: 80 (80), Dst Port: 2488

(2488), Seq: 422, Ack: 3024, Len: 256
Source Port: 80 (80)
# 源端口号
Destination Port: 2488 (2488)
# 目标端口号
[Stream index: 16]
[TCP Segment Len: 256]
Sequence number: 422    (relative sequence number)
# 序列号（相对序列号）
[Next sequence number: 678    (relative sequence number)]
# 下一个序列号（相对序列号）
Acknowledgment number: 3024    (relative ack number)
# 确认序列号
Header Length: 20 bytes
# 头部长度
.... 0000 0001 1000 = Flags: 0x018 (PSH, ACK)
# TCP标记字段
Window size value: 162
# 流量控制的窗口大小
Checksum: 0x6fc4 [validation disabled]
# TCP数据段的校验和
```

筛选器的简单用法

Wireshark的筛选器可以让我们找出我们所希望进行分析的数据包。简单来说， 一个筛选器就是定义了一定的条件，用来包含或者排除数据包的表达式。如果在实 际的分析中，不希望看到某一些数据包，那么就可以通过筛选器来屏蔽掉它们， 从而只显示我们感兴趣的内容。但是这里需要注意的是，有些时候，我们在进行网 络分析时，可能会因为设置了不恰当的筛选器而漏掉了一些关于这个网络情况的关 键数据，所以筛选器有时候也是一把双刃剑。

筛选器主要是在处理大量的数据时使用，比如我们刚才就使用了筛选器来专门显 示HTTP协议的数据包。这是关于筛选器的最简单的一种用法。筛选器也支持与 （and）、或（or）、非（not）等逻辑运算符，可以加强筛选的效率。比如我 们可以通过如下语句来显示出所有IP地址为180.97.34.134，端口号为80的数据包 ：

```
ip.addr==180.97.34.134 and tcp.port==80
```

或者也可以查看所有非TCP协议的协议：

```
!tcp
```

除了逻辑操作符，在筛选器中还可以使用比较操作符，比如等于（==）、不等于 （!=）、大于（>）、小于（<）、大于等于（>=）以及小于等于（<=）等。比如我 们现在只想查看长度小于等于150字节的数据包，则可以使用如下命令：

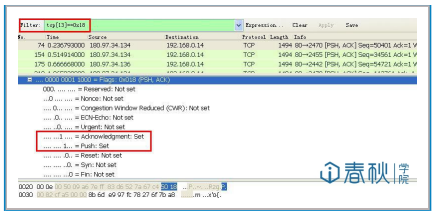
```
frame.len<=150
```

最后再讲一下Wireshark的协议域筛选器。这个功能可以帮助我们通过检查协议 头中的某一个字节来创建筛选条件，或者也可以匹配一个数据包中从某一特定位置 开始一定数量的字节。举例来说，比如我们想捕获带有(PSH,ACK)标志的TCP数据 包 。那么就可以检测TCP协议中偏移为13的标志位的情况。尽管这个标志位只有1 个字节，但是这个字节中的每一个比特位都是一个标志。

我们可以用如下命令进行查看：

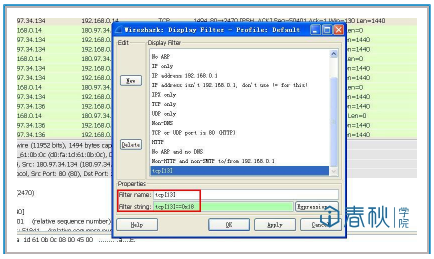
```
tcp[13]==0x18
```

显示结果如下：



如果我们在实际的分析过程中发现某一个筛选条件使用的比较频繁，那么可以考 虑将这个筛选条件保存下来，避免每次分析中的重复输入。我们可以在菜单栏中选 择“**Analyze**”->“**Display Filter**”，打开Display Filter对话框。

单击左边的New按钮，创建一个新的筛选器。并且在“**Filter Name**”中给 筛选器起一个名字，在“**Filter String**”中输入筛选表达式，之后单 击“**OK**”进行保存即可：



灵活使用筛选器会起到事半功倍的效果，我们在之后的分析中也会多次用到。