B.1 数据包表现形式

数据包能够以很多表现形式被解析。数据包原始数据可以表现为二进制数据—二进制的 0、1 序列，如：

```
   01100000010100110101110000001010110000010000000000010000000000
11010101101110000000000000000000000000000000000000000000100000100000
01101000000000000010000001100000000000000000000000010000001000000000
```

二进制数码是数据信息在最底层的表现形式，1 表示高电平信号，0 表示低电平信号。每一个数字是一位，八位是一字节。然而，人们很难阅读和理解二进制数据，所以我们通常将二进制数据转换为十六进制—由字母和数字组成的十六进制数据，如：

```
   4500 0034 40f2 4000 8006 535c ac10 1080
4a7d 5f68 0646 0050 7c23 5ab7 0000 0000
8002 2000 0b30 0000 0204 05b4 0103 0302
0101 0402
```

十六进制（通常简写为 hex）是使用数字 0～9 和字母 A～F 表示数值的计数系统。十六进制是常用的数据包表示形式，因为其形式简洁，并且很容易被转换为更基础的二进制。在十六进制中，两个字符表示一个字节，即八位。字节中的每个字符是一个半字节（4 位），左侧的值是高位（半）字节，右侧的值是低位（半）字节。在示例数据包中，第一个字节为 45，其中，高位字节是 4，低位字节是 5。

数据包中，字节的地址（或位置）使用偏移量表示法表达，从 0 开始。因此，数据包中的第一个字节（45）位于 0x00，第二个字节（00）位于 0x01，第三个字节（00）位于 0x02，依此类推。0x 说明使用的是十六进制表示法。在表示一个长度大于一字节的地址时，地址占用的字节数在一个冒号之后使用数值表示。例如，在表示示例数据包的前 4 字节的地址时，使用 0x00:4 表示。在我们后续使用数据包结构图剖析「驾驭一个神秘的数据包」中的未知协议时，此处的说明非常重要。

> 注意
>
> 据我观察，人们在分析数据包时普遍的错误是忘了从 0 开始计算地址。的确很难习惯从 0 开始计算，因为大部分人学习的都是从 1 开始计数。我已经分析数据包很多年了，但是仍然会犯这样的错。我能给出的最好的建议是，不要害怕掰手指数数。你也许会觉得这样看起来很愚蠢，但是这其实没什么丢人的，尤其是在能帮你得出正确答案的情况下。

在更高的层级，一个与 Wireshark 类似的工具使用协议分析器将数据包使用完全解析的形式展示出来，我们后续将介绍此工具。本例中的数据包被 Wireshark 解析后，如图 B-1 所示。

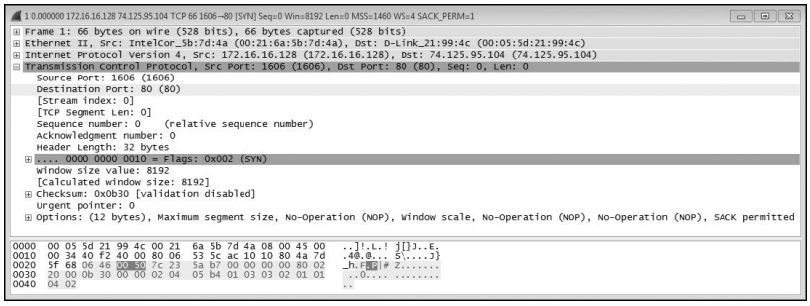

图 B-1　Wireshark 解析后的数据包

Wireshark 显示数据包信息，并为各字段添加标签。原始数据包并不包含标签，但是其中的数据按照协议标准规定的明确格式排列。完全解析数据包意味着将数据按照协议标准分析为带有标签的、可阅读的文本。

Wireshark 及类似的工具能够完全解析数据包，因为它们的内置协议分析器对协议各字段的地址、长度和值进行了定义。例如，图 B-1 中的数据包按照传输控制协议（TCP）标准分成多个部分，包括带有标签的字段和值。其中一个标签是源端口（Source Port），值为十进制的 1606。这使你在分析数据包时能够轻易找到指定信息。当你能够使用此类工具时，它们会成为你完成分析工作的一个高效的方式。

Wireshark 有成百上千个协议分析器，但是你仍有可能遇到 Wireshark 无法解析的协议；厂商定制的未广泛使用的协议和定制的恶意软件协议经常会是这种情况。当这样的事情发生时，数据包中只有部分内容能够被解析。这也是 Wireshark 默认在界面下方提供原始的十六进制包数据的原因（如图 B-1 所示）。

更普遍的情况是，如 Tcpdump 的命令行程序不提供太多的协议分析器，而是显示大量原始十六进制数据。对于一些更复杂的应用层协议而言，这种情况尤为常见，因为这类协议很难解析。因此，当我们使用 Tcpdump 时，看到被部分解析的数据包是常态。一个使用 Tcpdump 分析数据包的例子如图 B-2 所示。

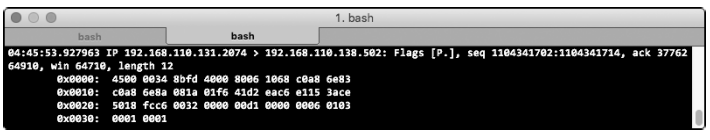

图 B-2　Tcpdump 中部分解析的数据包

当面对部分解析的数据包时，你需使用更底层的数据包结构知识。Wireshark、Tcpdump 及大部分其他工具提供了十六进制的原始包数据，

帮助我们进行更底层的分析工作。