6.4 控制输出

使用命令行工具的另一个优点是可以自定义输出。一般 GUI 应用会把所有的信息都告诉你，然后你可以自行寻找所需的内容。命令行工具通常只会显示最简输出，并强制你使用额外的命令参数来挖掘更高级的用法，TShark 和 Tcpdump 也不例外。默认情况下它们只会为一个数据包显示一行输出。如果你想看到协议细节或者单独字节这些更深的内容，就需要使用额外的命令参数了。

在 TShark 的输出里，每一行代表一个数据包，每一行输出的格式取决于数据包使用的协议类型。TShark 底层使用和 Wireshark 一样的解析器来分析数据包，所以 TShark 的输出和 Wireshark 的包列表窗口很像。正因为 TShark 可以解析七层协议，所以它能够比 Tcpdump 提供更多的有关包头信息的内容。

Tcpdump 中每行也代表一个数据包，根据不同的协议来规范每行的输出格式。因为 Tcpdump 不依赖于 Wireshark 的协议解析器，所以第 7 层的协议信息无法被解码。这也是 Tcpdump 的最大限制之一。取而代之的是，Tcpdump 单行数据包只会根据传输层协议（TCP 或 UDP）进行解码（我们会在第 8 章里着重讲解）。

TCP 包使用以下格式：

```
   [Timestamp] [Layer 3 Protocol] [Source IP].[Source Port] > [Dest
[Destination Port]: [TCP Flags], [TCP Sequence Number], [TCP Acknow
Number], [TCP Windows Size], [Data Length]
```

UDP 包使用以下格式：

```
   [Timestamp] [Layer 3 Protocol] [Source IP].[Source Port] > [Dest
[Destination Port]: [Layer 4 Protocol], [Data Length]
```

这种简单的单行总结对快速分析很有帮助，但最终你还是要对一个数据包进行深入分析。在 Wireshark 中，你会在包列表窗口里选择一个数据包，它将在下方的包细节和包字节窗口显示一些细节内容。使用命令行的命令也可以达到类似效果。

一个获取更多细节的简单方法是增加输出的冗余程度。

在 TShark 中，使用大写的 V 来增加冗余：

```
   C:\Program Files\Wireshark>tshark –r packets.pcap –V
```

这会提供类似 Wireshark 打开 packets.pcap 后包细节窗口里的内容。这里展示了一具具有正常冗余（基本总结）和扩展冗余（使用-V 参数获取）的包的示例。

首先正常输出：

```
   C:\Program Files\Wireshark>tshark -r packets.pcap -c1
  1   0.000000 172.16.16.172 -> 4.2.2.1      ICMP Echo (ping) reque
id=0x0001, seq=17/4352, ttl=128
```

现在使用更大的冗余选项来显示更多的内容：

```
   C:\Program Files\Wireshark>tshark -r packets.pcap -V -c1
Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
interface 0
    Interface id: 0 (\Device\NPF_{C30671C1-579D-4F33-9CC0-73EFFFE85
    Encapsulation type: Ethernet (1)
    Arrival Time: Dec 21, 2015 12:52:43.116551000 Eastern Standard
     [Time shift for this packet: 0.000000000 seconds]
--snip--
```

在 Tcpdump，小写的 v 是用来增加冗余的。这点跟 TShark 略有不同，Tcpdump 允许每个数据包显示不同层级的冗余信息。你可以通过增加 v 参数的数量来增加显示层级，至多到第 3 层，如下所示：

```
   sanders@ppa:~$ tcpdump –r packets.pcap –vvv
```

下面是在相同数据包下，Tcpdump 使用默认冗余选项和更高一级的冗余选项之间的比较。即便使用最大冗余选项，输出的信息也很难达到 TShark 那样的丰富度。

```
   sanders@ppa:~$ tcpdump -r packets.pcap -c1
reading from file packets.pcap, link-type EN10MB (Ethernet)
13:26:25.265937 IP 172.16.16.139 > a.resolvers.level3.net: ICMP ech
id 1759, seq 150, length 64
sanders@ppa:~$ tcpdump -r packets.pcap -c1 -v
reading from file packets.pcap, link-type EN10MB (Ethernet)
13:26:25.265937 IP (tos 0x0, ttl 64, id 37322, offset 0, flags [DF]
ICMP (1), length 84)
    172.16.16.139 > a.resolvers.level3.net: ICMP echo request, id 1
150, length 64
```

可以显示出多少细节取决于你当前分析数据包的协议类型。虽然高冗余级别是有用的，但是有时也很难让我们看清所有的内容。TShark 和 Tcpdump 储存了每个包的所有内容，你可以以十六进制字节或它的 ASCII 表示形式来查看。

在 TShark 里，你可以使用-x 参数来查看数据包的 ASCII 形式或十六进制字节形式，同时结合-r 参数把捕获文件读取到 TShark 里并显示出来。

```
C:\Program Files\Wireshark>tshark –xr packets.pcap
```

显示结果很像 Wireshark 的包字节窗口，如图 6-1 所示。

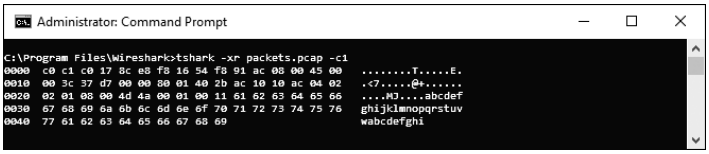

图 6-1　在 TShark 里以十六进制字节形式或 ASCII 形式表示原始数据包

类似地在 Tcpdump 里，你可以使用-X 参数，来查看数据包的 ASCII 形式或十六进制字节形式，同时结合-r 参数把捕获文件读取到 Tcpdump 里并显示出来，就像这样：

```
sanders@ppa:~$ tcpdump –Xr packets.pcap
```

这条命令的输出结果如图 6-2 所示。

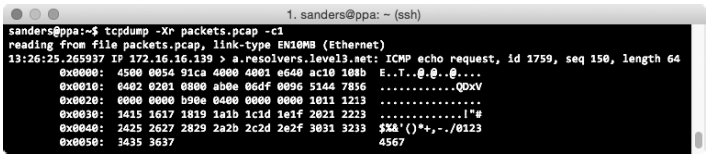

图 6-2　在 Tcpdump 里以十六进制字节形式或 ASCII 形式表示原始数据包

如果你需要的话，Tcpdump 还允许你获得更多的粒度。你可以使用-x（小写）参数只查看十六进制输出或者使用-A 参数只输出 ASCII 形式。

如果你添加了这些增加冗余的选项，则当数据输出在屏幕上飞快滚动时你会容易感到眼花缭乱。我以为，要做到最有效率的分析就要在命令行使用最少的信息显示你最关心的内容。我建议从默认的输出格式开始，当你有特别的包需要深入分析时，再使用更详细的输出选项。这种策略会避免你被大量数据所淹没。