6.6 应用过滤器

TShark 和 Tcpdump 的过滤器是非常灵活的，因为它们都遵从 BPF 捕获过滤器语法。TShark 也可以使用 Wireshark 的显示过滤器表达式。就像 Wireshark 一样，TShark 的捕获过滤器可以边捕获边过滤，也可以在捕获完成后过滤显示结果。我们从 TShark 的捕获过滤器开始讲起。

使用-f 参数来应用捕获过滤器，在双引号内请遵从 BPF 的语法。下面这条命令仅会抓取和储存目的端口号是 80 的 TCP 流量：

```
C:\Program Files\Wireshark>tshark –ni 1 –w packets.pcap –f "tcp port 80"
```

使用-Y 来应用显示捕获器，请在双引号内使用 Wireshark 的过滤器语法。在抓取流量的过程中，你可以使用像下面的命令：

```
C:\Program Files\Wireshark>tshark –ni 1 –w packets.pcap –Y "tcp.dstport == 80
```

使用类似的命令显示过滤器可以应用在已经捕获的文件中。以下命令会显示 packets.pcap 中所有符合过滤表达式的包：

```
C:\Program Files\Wireshark>tshark –r packets.pcap –Y "tcp.dstport == 80"
```

在 Tcpdump 中你可以在单引号里构造过滤表达式，然后附到命令的最后。以下的命令依然会捕获和存储目的端口号是 80 的 TCP 流量：

```
sanders@ppa:~$ tcpdump –nni eth0 –w packets.pcap 'tcp dst port 80'
```

当读取捕获文件时你也可以构造过滤器。以下命令会显示 packets.pcap 中所有符合过滤表达式的包：

```
sanders@ppa:~$ tcpdump –r packets.pcap 'tcp dst port 80'
```

需要牢记的一点是，如果没有在抓包的时候指明过滤器，那么你的捕获文件里通常会含有其他数据包。读取这个捕获文件后，你仅仅在屏幕上限制了所打出来的内容。

那么如果你有一个包含大量各种类型数据包的捕获文件，而你又想把需要的数据包过滤出来另存为一个文件，这时候怎么办呢？你可以结合使用-w 和-r 参数来解决：

```
sanders@ppa:~$ tcpdump –r packets.pcap 'tcp dst port 80' –w http_packets.pcap
```

这个命令会先读取 packets.pcap，过滤出目的 TCP 端口为 80 的数据包（http 用的端口），最后把这些数据包写入一个名叫 http_packets.pcap 的新文件里。当你既想把大型原文件.pcap 保存起来，又想在某时专注于分析其中一小部分时，这是个很常见的技巧。我经常使用这个技巧，特别是当我要把很大的捕获文件用 Tcpdump 切小，然后再放到 Wireshark 里分析时。毕竟小文件更加容易处理。

除了在一行命令后面直接加上过滤表达式，Tcpdump 还允许你指定一个包含一系列过滤器的 BPF 文件。这在有些情况下十分方便，特别是当你要应用一个极其复杂的过滤器表达式，且长度不能和 Tcpdump 的命令保持在同一行时。你可以使用-F 参数来指派一个 BPF 过滤器文件，就像这样：

```
sanders@ppa:~$ tcpdump –nni eth0 –F dns_servers.bpf
```

如果你的 BPF 文件太大，那么你也许会加一些注释，以帮助你理解每个部分的过滤表达式的功能和结构。值得注意的是，在 BPF 文件里直接加注释是非法的，如果不是 BPF 语法的话就会报错。但又因为注释对于解密大型 BPF 文件是非常有帮助的，所以我通常会使用两份 BPF 文件，一份不包含任何注释，是载入到 tcpdump 里的；另一份含有注释以供参考。