11.2.3 TCP 滑动窗口实战

看完 TCP 滑动窗口的理论之后，我们将在捕获文件 tcp_zerowindowrecovery. pcap 中探究它。

在这个文件中，我们从 192.168.0.20 发送给 192.168.0.30 的几个 TCP ACK 数据包开始。我们主要对 Windows Size 域感兴趣，可以在 Packet List 面板的 Info 列以及 Packet Details 面板的 TCP 头部看到它。从图 11-16 可以立即发现，在前面的 3 个数据包中，这个域的值不断减小。

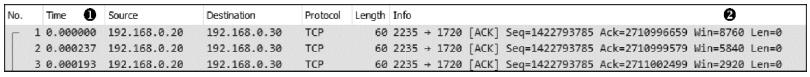

图 11-16　这些数据包的窗口大小在递减

这个值从第一个数据包的 8760 字节减少到第二个数据包的 5840 字节，接着又减为第三个数据包的 2920 字节 ❶。窗口大小值递减是主机延迟增加的典型指标。注意一下 Time 列的信息，这是在极短的时间内发生的 ❷。当窗口大小像这样快速减小时，它很可能会减至零，如图 11-17 所示，数据包 4 正是这样的情形。

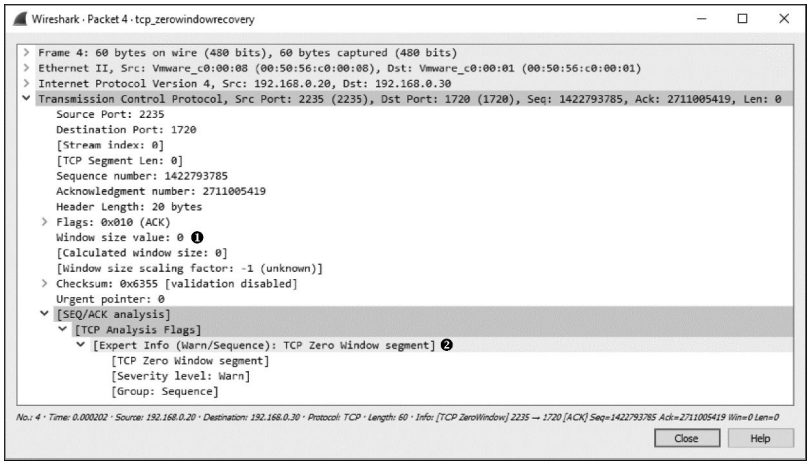

图 11-17　零窗口数据包说明了主机不能再接收任何数据

第 4 个数据包也是从 192.168.0.20 发往 192.168.0.30 的，但它的目的是告诉 192.168.0.30 它不能再接收任何数据。在 TCP 头部就可以看到这个数值 0❶，而且 Wireshark 也在 Packet List 面板的 Info 列以及 TCP 头部 SEQ/ACK Analysis 部分，告诉我们这是一个零窗口数据包 ❷。

一旦收到零窗口数据包，192.168.0.30 这个设备就不再发送任何数据，直到它从 192.168.0.20 收到一个窗口更新，通知它窗口大小已经增长了为

止。幸好，在这个捕获文件里，导致零窗口的问题是暂时的。因此，如图 11-18 所示，发送的下一个数据包就是窗口更新。

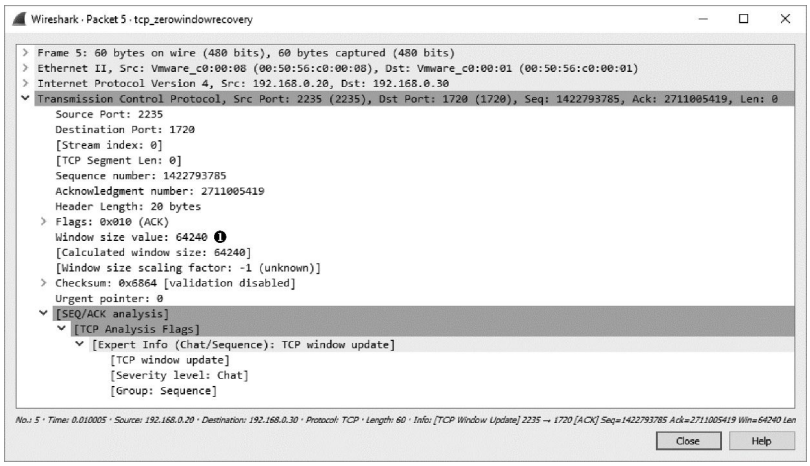

图 11-18　TCP 窗口更新数据包告诉其他主机它又可以传输数据了

在这个例子中，窗口大小增长到了非常健康的 64240 字节 ❶。Wireshark 再一次在 SEQ/ACK Analysis 标题下面告诉我们，这是一个窗口更新。

一旦收到这个更新数据包，192.168.0.30 主机就可以再次发送数据，如数据包 6 和 7 所示。这个过程非常迅速。就算它只是多持续一点点时间，也可能会引起网络「打嗝」，导致数据传输变慢或失败。

最后再看滑动窗口，查看一下 tcp_zerowindowdead.pcap 文件。捕获记录中的第一个数据包是从 195.81.202.68 发送到 172.31.136.85 的正常 HTTP 流量。如图 11-19 所示，紧接着就是一个从 172.31.136.85 返回的零窗口数据包。

这看起来跟图 11-17 里的零窗口数据包非常相似，但结果却很不相同。在图 11-20 中，我们并没有看到 172.31.136.85 主机发送使通信恢复的窗口更新，而是看到一个保活数据包。

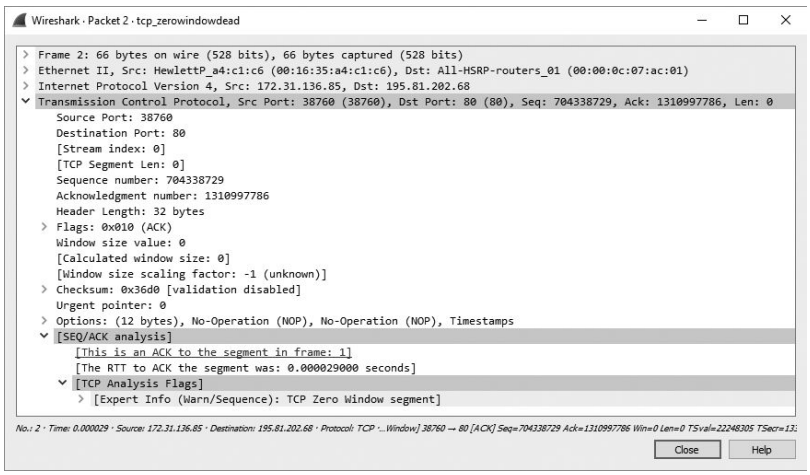

图 11-19　零窗口数据包使数据传输暂停

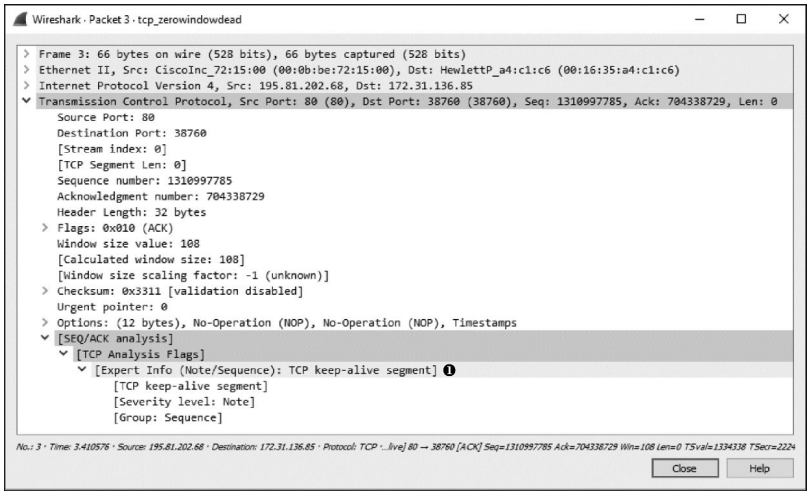

图 11-20　保活数据包保证零窗口主机仍然在线

Wireshark 在 Packet Details 面板中 TCP 头部的 SEQ/ACK Analysis 标题下，将这个数据包标记为保活数据包 ❶。我们从 Time 列可得知，在收到上一个数据包 3.4s 后，出现了这个数据包。如图 11-21 所示，这个过程又持续了几次：一台主机发送零窗口数据包，另一台则发送保活数据包。

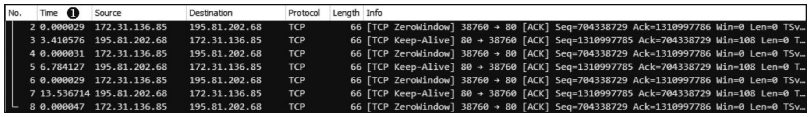

图 11-21　零窗口和保活数据包不断出现

这些保活数据包以 3.4s、6.8s、13.5s 的间隔出现 ❶。这个过程可能会持续相当长的时间，这取决于通信设备采用了哪个操作系统。在这个例子中，你可以发现，随着 Time 列数值的增长，连接暂停了将近 25s。想象一下，尝试向域控制器认证或者从网上下载文件时，25s 的延迟真是难以接受！