

10.6.2 分析

除了网络上的基本信息流量之外，我们完全不了解程序员开发的应用程序。捕获文件看起来是以一些 FTP 流量开始的，因此我们将调查这是不是传输 CSV 文件采用的机制。简洁、干净的通信最适合查看通信流量图了。选择 Statistics -> Flow Graph，然后单击「OK」。图 10-33 显示了结果图像。

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	172.16.16.121	TCP	66	2555 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.000071	172.16.16.121	172.16.16.128	TCP	66	21 → 2555 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	0.000242	172.16.16.128	172.16.16.121	TCP	60	2555 → 21 [ACK] Seq=1 Ack=1 Win=17520 Len=0
4	0.002749	172.16.16.121	172.16.16.128	FTP	96	Response: 220 FileZilla Server version 0.9.34 beta
5	0.002948	172.16.16.128	172.16.16.121	FTP	70	Request: USER salesxfer
6	0.003396	172.16.16.121	172.16.16.128	FTP	91	Response: 331 Password required for salesxfer
7	0.003514	172.16.16.128	172.16.16.121	FTP	69	Request: PASS p@ssw0rd
8	0.004862	172.16.16.121	172.16.16.128	FTP	69	Response: 230 Logged on

图 10-33 流量图提供了 FTP 通信的快速视图

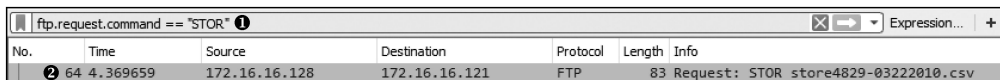
首先观察一下包列表（见图 10-34），我们会看到一个 172.16.16.128①与 172.16.16.121②之间的基本 FTP 连接。由于 172.16.16.128 发起连接，因此我们猜测它是客户端，172.16.16.121 则是汇总与处理数据的服务器。流量图确认这些流量只用到了 FTP 协议，在握手竞争之后，我们开始看到来自客户端的 FTP 请求和来自服务器的响应 ③。

我们知道这里会有一些数据传输，因此我们用 FTP 的知识，定位开始传输数据的位置。FTP 连接和数据传输是由客户端发起的，因此我们应该寻找用于上传数据到 FTP 服务器的 FTP STOR 命令。简单的方法是生成一个过滤器。

这个捕获文件里到处都是 FTP 请求命令，因此我们不需要面对表达式生成器中的数百个协议和选项，只要在 Packet List 面板中直接生成过滤器就行了。为此，我们首先需要选择一个出现有 FTP 请求命令的数据包。我们选择了数据包 5，这是最接近列表顶部的一个。然后展开 Packet Details 面板的 FTP section 和 USER section。右击 Request Command: USER 域，并选择 Prepare a Filter。最后，选择 Selected。

这将生成一个筛选含有 FTP USER 请求命令的数据包的过滤器，并出现在过滤器对话框中。接着，如图 10-34 所示，编辑过滤器，将单词 USER 替换成 STOR ①。

现在按回车键应用这个过滤器，你会看见捕获文件里只有一个 STOR 命令，在数据包 64 ②中。



No.	Time	Source	Destination	Protocol	Length	Info
64	4.369659	172.16.16.128	172.16.16.121	FTP	83	Request: STOR store4829-03222010.csv

图 10-34 这个过滤器有助于识别数据从哪里开始传输

既然我们已经知道数据从哪里开始传输了，就可以单击 Packet List 面板上方的 Clear 按钮清除过滤器。

查看从数据包 64 开始的捕获文件，我们看见这个数据包指定传输 store4829-03222010.csv 文件 ❶，如图 10-35 所示。

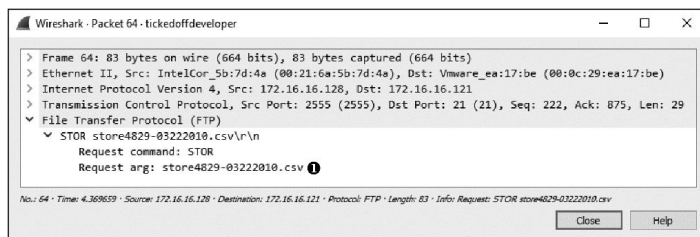


图 10-35 使用 FTP 传输 CSV 文件

STOR 命令后面的数据包使用了不同的端口，但它们被识别成 FTP 数据传输的一部分。我们已经验证数据在传输了，但我们仍然没有证明程序员是错的。为此，我们需要从捕获的数据包中提取传输文件，以展示文件在网络中传输后并没有被损坏。

当文件以未加密格式在网络中传输时，它会被分解成多个段，并在目的地被重新组装。在这个场景中，我们在数据包到达目的地并且尚未被组装之时捕获它们。数据就在那儿了，我们只需要将文件提取为数据流来重新组装它。为此，选择 FTP 数据流的任一个数据包（比如数据包 66），并单击 Follow TCP Stream。结果显示在 TCP 流中，如图 10-36 所示。

由于数据在 FTP 中以明文传输，所以我们能看到它，但却不能仅由此断定文件是完整的。为了重组数据，以便将其提取为原始格式，我们单击 Save As 按钮并指定数据包 64 显示的文件名，如图 10-37 所示。然后单击 Save。

保存操作的结果应该是一个 CSV 文件，这是对商店系统传过来的文件的字节层次的复制。我们通过比较原始文件和提取文件的 MD5 哈希值来验证该文件。MD5 哈希值应该是一样的，如图 10-37 所示。

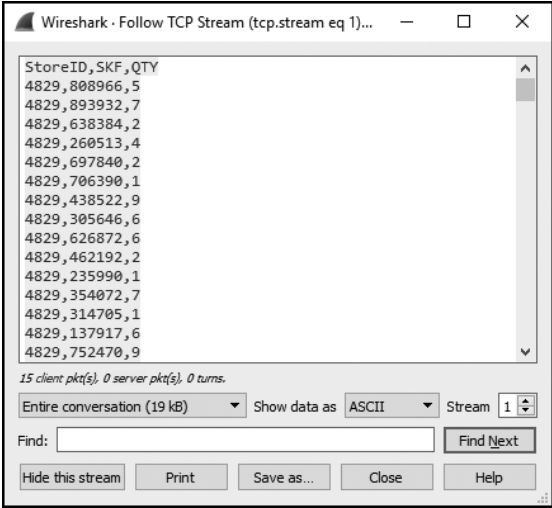


图 10-36 TCP 流显示传输的数据将数据流保存为原始文件名

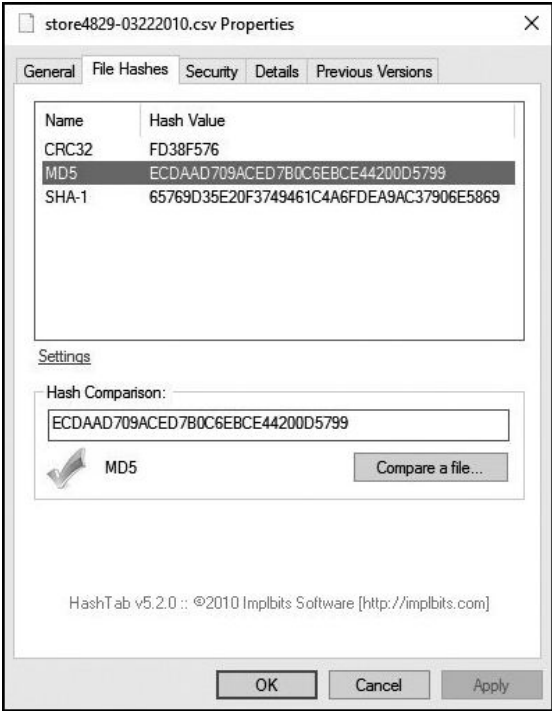


图 10-37 原始文件和提取文件的 MD5 哈希值相等

通过比较文件，我们可以证明网络并不是应用程序数据库出错的原因。文件从商店传输到收集服务器时是完整的，所以肯定是应用程序处理文件时出错了。