9.4.2 跟踪一封电子邮件

对电子邮件如何传输有了基本理解之后，我们开始查看这一过程中的数据包。首先从图 9-29 所描述的场景开始。

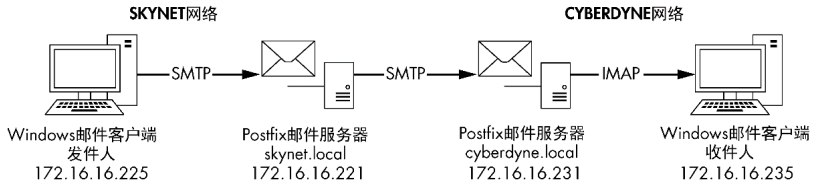

图 9-29　跟踪一封电子邮件从发送方至收件方的传输过程

以上场景分为 3 个步骤。

（1）用户从计算机（172.16.16.225）发送一封电子邮件，邮件客户端使用 SMTP 协议，将邮件传输至本地邮件服务器（172.16.16.221/skynet.local 域）。

（2）本地邮件服务器接收邮件，并使用 SMTP 将其传输至远程邮件服务器（172.16.16.231/cyberdyne.local 域）。

（3）远程邮件服务器接收邮件，并将其与适当的邮箱相关联。收件方计算机（172.16.16.235）上的邮件客户端通过 IMAP 协议取回邮件。

第 1 步：客户端至本地邮件服务器

我们从第一步开始逐步查看邮件发送过程，对应的数据包文件为 mail_sender_client_1.pcapng。这个文件从用户在邮件客户端上单击「发送」按钮开始，用户计算机和本地邮件服务器在前 3 个数据包中完成 TCP 握手。

> 注意
>
> 本节中，你可以忽略抓包文件中所有的「ETHERNET FRAME CHECK SEQUENCE INCORRECT」错误。在实验环境中生成这些数据包时的一些人工操作导致了这些错误。

连接建立后，客户端使用 SMTP 将邮件传输至邮件服务器。分析数据包时，你可以逐个浏览 SMTP 请求并回复数据包，在「数据包详情」窗口中查看 SMTP 部分，但是有一种更简便的方式。由于 SMTP 是一种简单的传输协议，并且在这个例子中通信过程为明文传输，因此你可以跟踪 TCP 流，

从而在一个窗口中查看整个传输过程。在抓包文件中任意选择一个数据包，然后右键选择「Follow > TCP Stream」。传输的 TCP 流如图 9-30 所示。

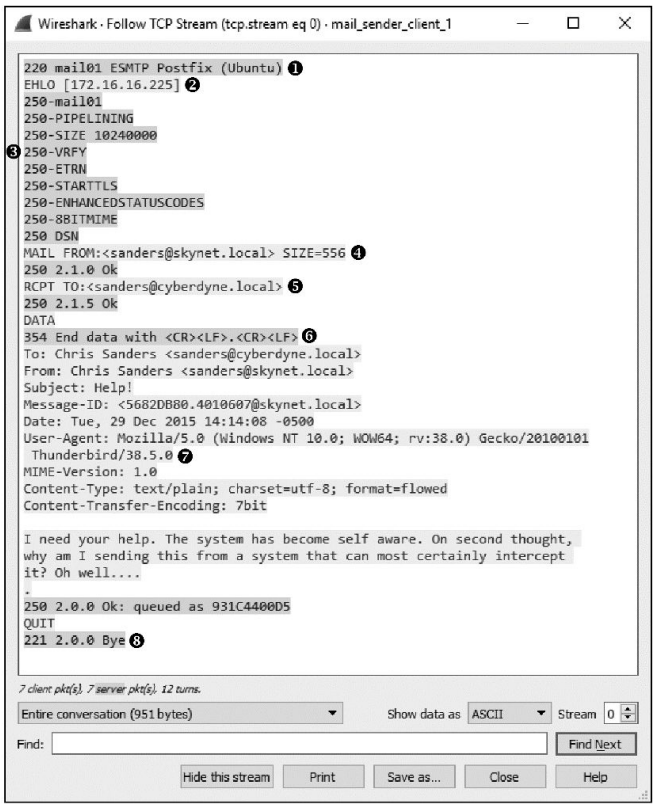

图 9-30　查看从邮件客户端至邮件服务器的 TCP 流

在连接建立后，邮件服务器在数据包 4 向客户端发送一个服务标签，表明此服务器已经准备好接收命令。在本例中，服务器显示为运行在 Ubuntu Linux 系统上的 Postfix 服务器 ❶。同时，此服务器支持接收扩展 SMTP（ESMTP）命令。ESMTP 是 SMTP 的扩展，允许在邮件传输中使用更多的命令。

邮件客户端在数据包 5 中通过 EHLO 命令进行回复 ❷。当邮件服务器支持 ESMTP 时，EHLO 作为「Hello」命令，让服务器识别发信的客户端主机。在不支持 ESMTP 的情况下，使用 HELO 命令进行发信客户端识别。在本例中，发信方使用 IP 地址作为识别方式，此外，域名也可以作为识别方式。

在数据包 7 中，服务器回复内容包括 VRFY、STARTTLS 和 SIZE 10240000❸。这些内容表示此 SMTP 服务器支持的命令，用于告知客户端在邮件传输过程中可用的命令范围。在发送邮件之前，这个特征协商过程会发生在每次 SMTP 传输的开始部分。邮件传输从数据包 8 开始，这个抓包文件剩余的大部分内容均为邮件传输过程的数据包。

SMTP 由客户端发送的简单命令和参数值控制，服务器会回复相应的响应码。这种设计是为了协议简化，与 HTTP 和 TELNET 很相似。数据包 8 和数据包 9 是一组示例请求/回复，客户端发送了 MAIL 命令，参数为

FROM:<sanders@ skynet.local> SIZE=556❹；服务器回复的响应码为 250（请求的邮件操作完成），参数为 2.1.0 Ok。在这次请求中，客户端发送了发信方的邮箱地址和邮件大小，服务器响应说明数据已经被接收，并且格式正确。类似的传输过程也发生在数据包 10 和数据包 11；客户端发送 RCPT 命令，参数为 TO:<sanders@cyberdyne. local>❺，服务器响应为 250 2.1.5 Ok。

剩余的部分是传输邮件内容数据。如数据包 12 所示，客户端使用 DATA 命令启动邮件内容传输过程。服务器响应码为 354，并附带信息 ❻，其中，响应码 354 表示服务器为这封邮件创建了缓冲区，客户端可以开始邮件传输；附带信息表示客户端需要发送一个 <CR><LF>.<CR><LF> 用于标记传输终止。这封邮件使用明文传输，并且响应码表示传输成功。你会注意到，邮件文本中包含一些附加信息，包括日期、内容类型和编码，以及传输使用的用户代理。以上信息表明，发信用户使用 Mozilla Thunderbird 发送了这封邮件 ❼。

传输完成后，在数据包 18 中，邮件客户端使用不带参数的 QUIT 命令，终止了 SMTP 连接。在数据包 19 中，服务器回复响应码 221（＜domain＞ 传输通道关闭），附带参数为 2.0.0 Bye❽。TCP 连接在数据包 20～23 中正常关闭。

第 2 步：本地服务器至远程服务器

接下来，我们将从 skynet.local 中的本地邮件服务器（地址为 172.16.16.221）的角度，考虑这个邮件传输场景。用于分析的数据包存储在 mail*sender_server*2.pcapng 文件中，这些数据包可以直接在邮件服务器上抓取。由于邮件从客户端至本地邮件服务器的传输过程，也被邮件服务器上的抓包工具记录，因此你会发现，mail2_sender_server_2.pcapng 包含的前 20 多个数据包与第 1 步中的数据包一致。

如果邮件目的地是 skynet.local 域中的其他邮箱，那么我们不会发现其他的 SMTP 传输；我们将看到某个邮件客户端使用 POP3 或 IMAP 协议收取邮件。然而，由于这封邮件被发送至 cyberdyne.local 域，因此本地 SMTP 服务器必须将邮件传输至为 cyberdyne.local 提供服务的远程邮件服务器。在数据包 22 中，本地邮件服务器 172.16.16.221 和远程邮件服务器 172.16.16.231 开始进行 TCP 握手，并开始第 2 步传输过程。

> 注意
>
> 在现实场景中，邮件服务器通过一种特殊的 DNS 记录类型——邮件交换（MX）记录，定位其他的邮件服务器。由于本节的场景为实验室环境，远程邮件服务器的 IP 地址已经预配置在本地邮件服务器上，因此我们将不会看到 DNS 通信过程。在进行邮件发送故障检查时，需要考虑 DNS 问题及与邮件相关的特定协议的问题。

连接建立完成后，我们在数据包列表中看到，邮件传输至远程服务器的过程使用 SMTP 协议。跟踪传输过程的 TCP 流，能够更清晰地查看这个会话过程，如图 9-31 所示。如果需要将这个连接单独提取出来，请在过滤器中使用 tcp.stream==1 作为筛选条件。

图 9-31　查看从本地邮件服务器至远程邮件服务器的 TCP 流

这个传输过程与图 9-30 中的过程基本一致。本质上，两者传输的邮件内容一致，区别在于，步骤 2 中的传输过程发生在两个服务器之间，步骤 1 中的过程发生在客户端和服务器之间。远程服务器被标记为 mail02❶，本地服务器被标记为 mail01❷，两个服务器共享了一系列可用命令 ❸；邮件内容 ❹ 包括步骤 1 中完整的邮件文本，以及附加在 To: Chris Sanders <sander@cyberdyne. local> 行之前的部分。这个通信过程在数据包 27～35 中，以一个 TCP 断开作为结束。

从根本上说，服务器不关注邮件是来自于邮件客户端还是另一个 SMTP 服务器，所以客户端——服务器邮件传输中所有的规则和过程在服务器——服务器邮件传输过程中也都适用（任意形式的访问控制策略除外）。在实际情况中，本地邮件服务器和远程邮件服务器可能并不具备相同的特征参数集合（包括命令集和参数集），或是基于完全不同的平台。这也是 SMTP 要进行初始化通信的非常重要的原因；这一过程确保了在邮件传输之前，收件服务器将自身支持的命令和参数集发送给发件方，从而完成传输使用命令的协商。在 SMTP 客户端或服务器明确了收件服务器的特征参数后，能够保证发件方发送的 SMTP 命令被收件服务器识别，并且邮件被正常传输。这一机制

使 SMTP 能够在大量不同的客户端和服务器之间应用，同时，也让我们在发送邮件时，不需要了解收件方的网络设备信息。

第 3 步：远程服务器至远程客户端

现在，我们的邮件到达了远程邮件服务器，此服务器负责将邮件投递至 cyberdyne.local 中的邮箱。我们将查看在远程邮件服务器上抓取的数据包 mail_receiver_server_3.pcapng，如图 9-32 所示。

我们再次发现前 15 个数据包很熟悉，因为传输的邮件内容与前两个步骤相同，不同的是，源地址变为本地邮件服务器 ❶，目的地址变为远程邮件服务器 ❷。这个流程完成后，SMTP 服务将邮件和指定的邮箱进行关联，之后，预期的收件方可以通过邮件客户端收取邮件。

正如之前提到的，SMTP 主要被用来发送邮件；截至目前，SMTP 也是发送邮件时常用的协议。从服务器上的邮箱收取邮件的方式更加多样，同时，由于在收取邮件事件中不断有新的需求，因此出现了数种用于完成邮件收取任务的协议。其中，比较流行的是邮局协议 v3（POP3）和 Internet 邮件访问协议（IMAP）。在本例中，远程客户端使用 IMAP 从邮件服务器收取邮件，对应通信发生在数据包 16～34 中。

图 9-32　查看从本地邮件服务器至远程邮件服务器的 TCP 流

本书并不会涵盖 IMAP 的内容；在本例中，即使我们介绍了 IMAP，也并不会对数据包分析带来多大帮助——这个通信过程是加密的。查看数据包 21❶，你会发现客户端（172.16.16.235）向邮件服务器（172.16.16.231）发送了 STARTTLS 命令，如图 9-33 所示。

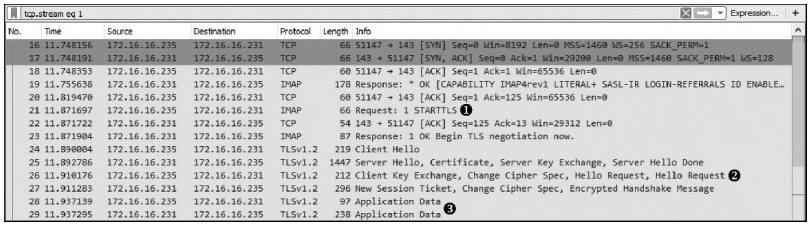

图 9-33　STARTTLS 命令表明此次 IMAP 通信将被加密

这个命令告知服务器，客户端将使用 TLS 加密收取邮件的过程。双方在数据包 24～27 中 ❷ 建立了安全信道，在余下的数据包中 ❸，邮件收取使用 TLS（传输层安全）协议完成。当查看这些数据包，或者尝试跟踪 TCP 流时（见图 9-34），你会发现内容不具备可读性，目的是防止邮件被恶意用户截获，避免数据劫持和嗅探的发生。

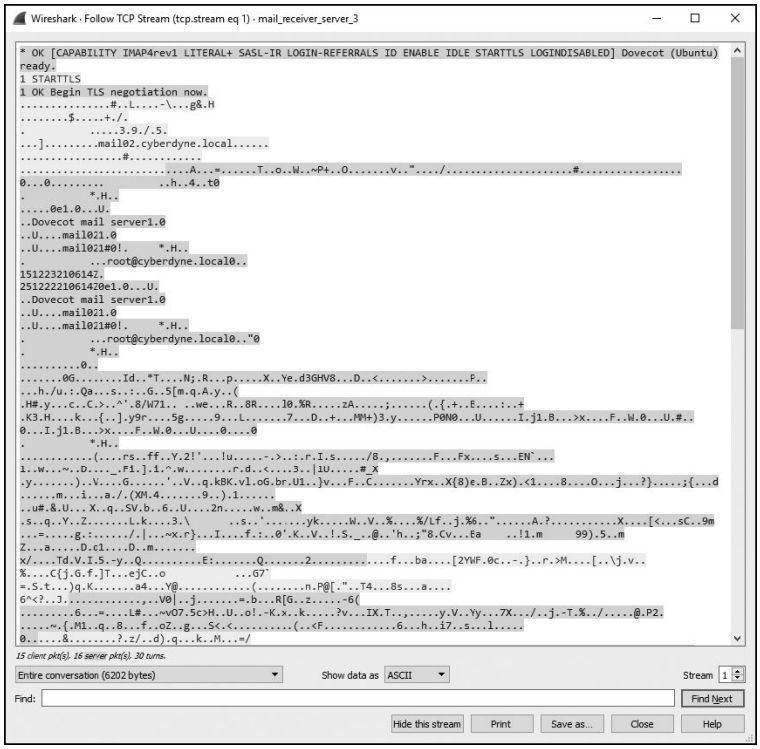

图 9-34　客户端下载邮件过程使用加密 IMAP 传输

最后，在这些数据包被接收后，不同域内两个用户之间的邮件发送过程就完成了。