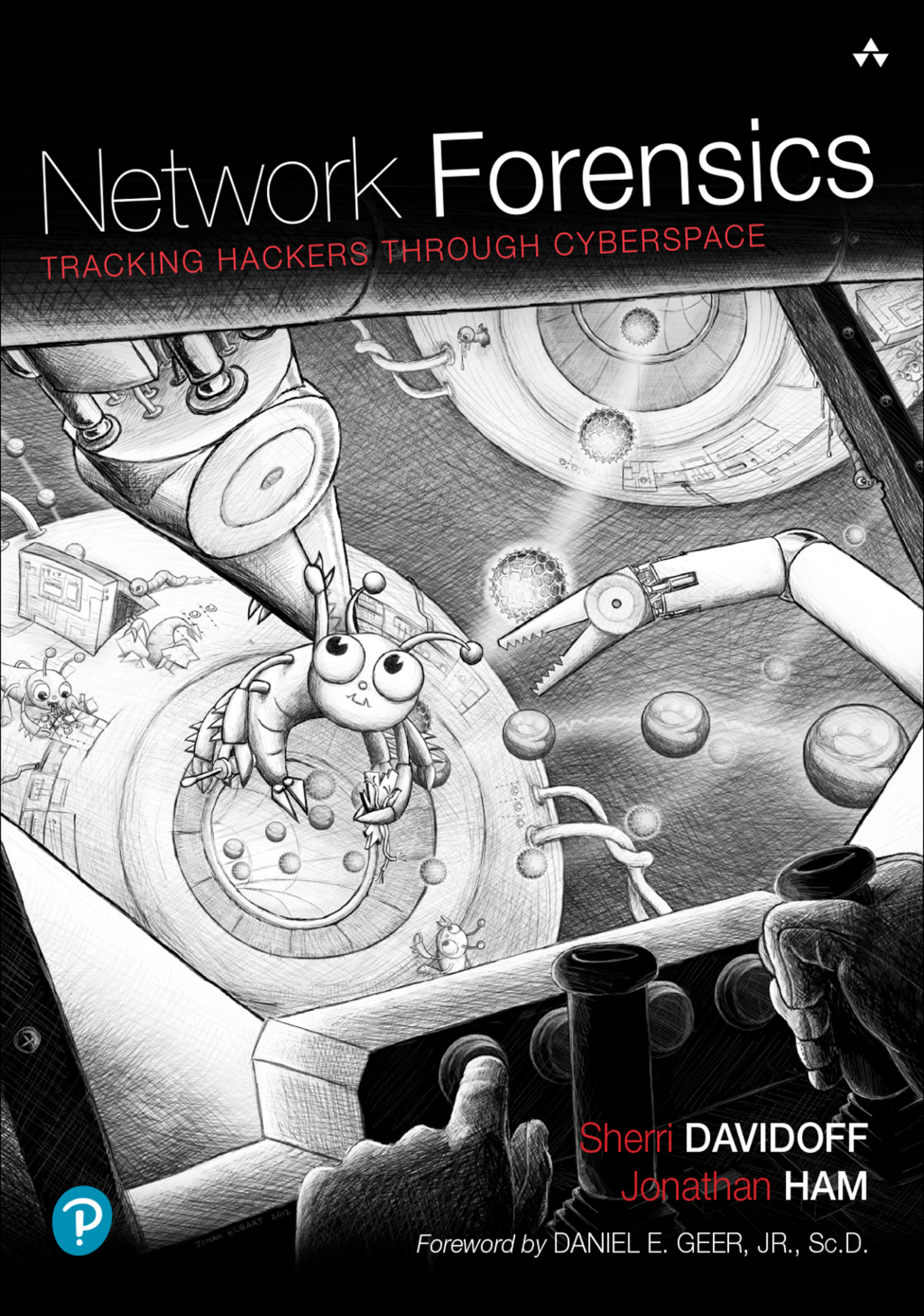
Network Forensics

TRACKING HACKERS THROUGH CYBERSPACE



Sherri DAVIDOFF
Jonathan HAM

Foreword by DANIEL E. GEER, JR., Sc.D.

Network Forensics

Finally, we see the body of the email, as shown after the "DATA" SMTP message. In this section, we can infer that the user sneakyg33ky@aol.com is representing herself as "Ann Dercover." The "Subject" of this email was "need a favor," and the message contained in the body was short but intriguing: "Hey, can you hook me up quick with that fake passport you were talking = about? - Ann." Could our suspect have been planning an unexpected, illicit trip abroad?

```
DATA
354 End data with <CR><LF>.<CR><LF>
Message-ID: <00ab01cc14c9$227de600$6b1ea8c0@annlaptop>
From: "Ann Dercover" <sneakyg33ky@aol.com>
To: <inter0pt1c@aol.com>
Subject: need a favor
Date: Tue, 17 May 2011 13:32:17 -0600
MIME-Version: 1.0
Content-Type: multipart/alternative;
.boundary="----=_NextPart_000_00A8_01CC1496.D700DE30"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2900.2180
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2180

This is a multi-part message in MIME format.

------=_NextPart_000_00A8_01CC1496.D700DE30
Content-Type: text/plain;
.charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

Hey, can you hook me up quick with that fake passport you were talking =
about? - Ann
...
```

In a similar fashion, we can use Wireshark's "Follow TCP Stream" function to examine the TCP stream containing the second SMTP packet we matched using ngrep, at 2011/05/17 13:34:16.481132. The results are shown in Figure 4–39. In this conversation, we see that the same user, sneakyg33ky@aol.com, sent an email to d4rktangent@gmail.com, with the "Subject" "lunch next week." In this email, the author writes:

```
Sorry-- I can't do lunch next week after all. Heading out of town. =
Another time!

-Ann
```

Apparently the sender's plans have changed!

4.6.5 SMTP Analysis—TCPFlow

Finally, let's examine the third matching SMTP packet from our ngrep results, created at 2011/05/17 13:35:16.962873. We could do this in the same way as the others, using Wireshark's "Follow TCP Stream," but instead let's practice using command-line tools (which typically scale better).

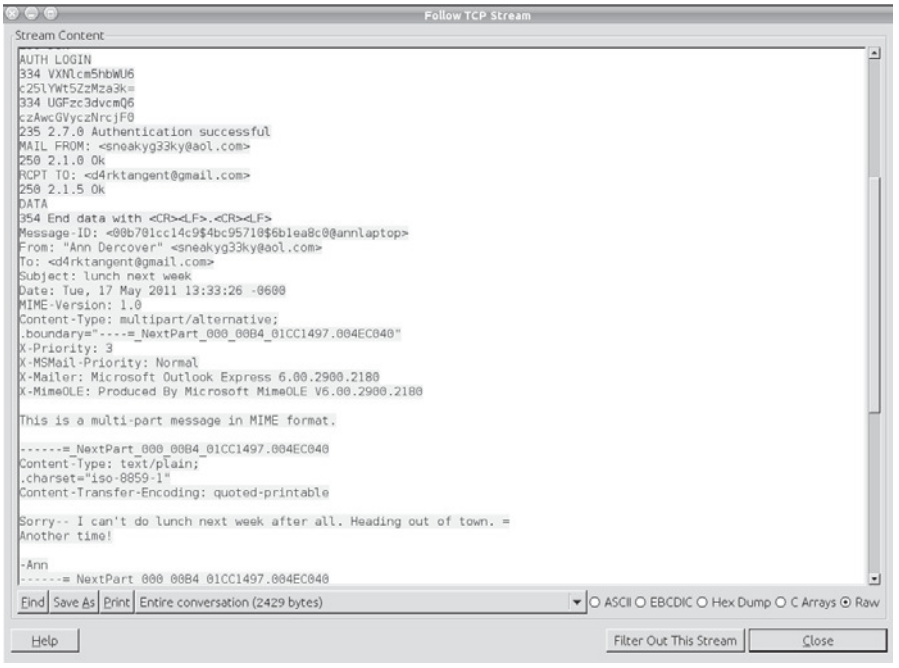


Figure 4–39. An SMTP conversation containing the second packet matched as part of our ngrep search.

Recall from our ngrep output that the matched TCP packet was sent from 192.168.30.108:1689 to 64.12.168.40:587 (the conversation that contained this packet was, of course, bidirectional). In the output below, we use tcpflow with an appropriate BPF filter to list all TCP flows between ports 1689 and 587. As you can see, tcpflow reconstructed two unidirectional data streams, one from 192.168.30.108:1689 to 64.12.168.40, and the other in the opposite direction.

```
$ tcpflow -v -r evidence-packet-analysis.pcap  'port 1689 and port 587'
tcpflow[333]: tcpflow version 0.21 by Jeremy Elson <jelson@circlemud.org>
tcpflow[333]: looking for handler for datalink type 1 for interface evidence-
    packet-analysis.pcap
tcpflow[333]: found max FDs to be 16 using OPEN_MAX
tcpflow[333]: 192.168.030.108.01689-064.012.168.040.00587: new flow
tcpflow[333]: 064.012.168.040.00587-192.168.030.108.01689: new flow
tcpflow[333]: 064.012.168.040.00587-192.168.030.108.01689: opening new output
     file
tcpflow[333]: 192.168.030.108.01689-064.012.168.040.00587: opening new output
     file
```

Let's examine the reconstructed content sent from 192.168.30.108 (Ann's computer) to the remote server 64.12.168.40. If the contents are indeed SMTP data as we expect, then we will be most interested in the outbound content from Ann's computer to the MSA, which

would contain the SMTP "MAIL FROM" and "RCPT TO" headers and authentication data, as well as the body of the email.

Viewing the outbound content reconstructed by tcpflow, we see the following:

```
$ less 192.168.030.108.01689-064.012.168.040.00587
EHLO annlaptop
AUTH LOGIN
c25lYWt5ZzMza3k=
czAwcGVyczNrcjF0
MAIL FROM: <sneakyg33ky@aol.com>
RCPT TO: <mistersekritx@aol.com>
DATA
Message-ID: <00bc01cc14c9$6fd1bc60$6b1ea8c0@annlaptop>
From: "Ann Dercover" <sneakyg33ky@aol.com>
To: <mistersekritx@aol.com>
Subject: rendezvous
Date: Tue, 17 May 2011 13:34:26 -0600
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="----=_NextPart_000_00B8_01CC1497.244B3EB0"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2900.2180
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2180

This is a multi-part message in MIME format.

------=_NextPart_000_00B8_01CC1497.244B3EB0
Content-Type: multipart/alternative;
        boundary="----=_NextPart_001_00B9_01CC1497.244B3EB0"


------=_NextPart_001_00B9_01CC1497.244B3EB0
Content-Type: text/plain;
        charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

Hi sweetheart! Bring your fake passport and a bathing suit. Address =
attached. love, Ann
------=_NextPart_001_00B9_01CC1497.244B3EB0
Content-Type: text/html;
...
```

In the SMTP conversation shown above, we see the outbound portion of a communication with a remote SMTP server. The "MAIL FROM" and "RCPT TO" SMTP messages shown below indicate that the email sender was sneakyg33ky@aol.com, and the recipient was mistersekritx@aol.com. In the body of the email, we can see that the sender has represented herself once again as "Ann Dercover." The "Subject" of this email was "rendezvous." As we saw above, the message in the body of the email was as follows:

```
Hi sweetheart! Bring your fake passport and a bathing suit. Address =
attached. love, Ann
```

As the message suggests, there is an attachment to the email. Scrolling further in the tcpflow output, we see the following section:

```
------=_NextPart_000_00B8_01CC1497.244B3EB0
Content-Type: application/octet-stream;
        name="secretrendezvous.docx"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
        filename="secretrendezvous.docx"

UEsDBBQABgAIAAAAIQCht/xGcgEAAFIFAAATAAgCW0NvbnRlbnRfVHlwZXNdLnhtbCCiBAIooAAC
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
...
```

Based on the line, "Content-Transfer-Encoding: base64," it appears that the attachment to the message is Base64-encoded, and the filename of the attachment is "secretrendezvous.docx."

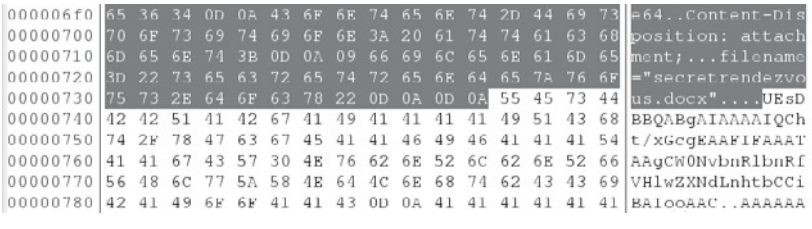
4.6.6 SMTP Analysis—Attachment File Carving

Now, let's carve the attachment out of the message body. To do this manually, open the stream dump in the Bless hex editor:

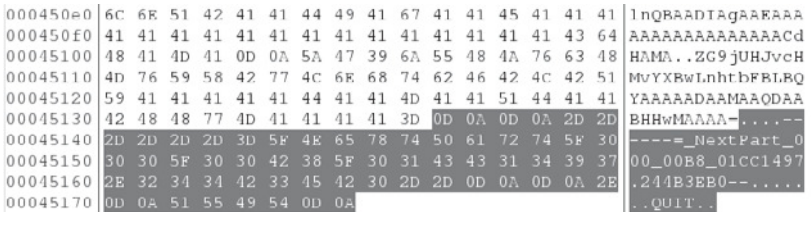
```
$ bless 192.168.030.108.01689-064.012.168.040.00587
```

Cut the SMTP and MIME protocol information off the top and bottom, as shown in Figure 4–40. The email contains multiple parts, as indicated by the mail headers:

```
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="----=_NextPart_000_00B8_01CC1497.244B3EB0"
```



(a)



(b)

Figure 4–40. This screenshot shows how we carve Ann's email attachment out of the "Ann Skips Bail" packet capture. The top image (a) shows us cropping the top part of the attachment, and the bottom file (b) shows us cropping the end of the attachment.

The attachment of greatest interest is contained in the part labeled:

```
------=_NextPart_000_00B8_01CC1497.244B3EB0
Content-Type: application/octet-stream;
        name="secretrendezvous.docx"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
        filename="secretrendezvous.docx
```

Immediately after these lines, there are TWO carriage-return/linefeeds (CRLF). (In hexadecimal, a carriage return is "0x0D" and a linefeed is "0x0A." Please see Section 10.5.2.1 for additional discussion of file carving using CRLF sequences as markers.) At the end of the message part, we see another sequence of TWO CRLFs. To carve out the attachment, we begin carving immediately after the first set of CRLFs and end just before the second set.[64] Let's save the carved file as "evidence-packet-analysis-smtp3-attachment."

The carved attachment contains some line breaks. We need to remove these in order to decode the Base64 encoding. For this purpose, we'll use the "fromdos" command by Christopher Heng, which is distributed as part of the "tofrodos" Debian package:

```
$ fromdos -b evidence-packet-analysis-smtp3-attachment
```

Now, we can decode the file and recreate the original attachment:

```
$ base64 -d evidence-packet-analysis-smtp3-attachment > secretrendezvous.docx
```

Check the file type:

```
$ file secretrendezvous.docx
secretrendezvous.docx: Zip archive data, at least v2.0 to extract
```

This makes sense, since .docx files are zip archive data. Let's take the cryptographic checksums:

```
$ md5sum secretrendezvous.docx
9049b6d9e26fe878680eb3f28d72d1d2  secretrendezvous.docx

$ sha256sum secretrendezvous.docx
24601c174587be4ddfff0b9e6d598618c6abfcfadb16f7dd6dbd7a24aed6fec8
    secretrendezvous.docx
```

4.6.7 Viewing the Attachment

Naturally, we want to view the contents of the attachment, which has been labeled with a .docx extension. Always remember that mainstream document-viewing programs often modify the files even when only used to view the file. When viewing a file, make sure that you are working on a *copy* of the evidence, and not the evidence itself. You can also use filesystem permissions, hardware write blockers, and other mechanisms to reduce the risk that the application will modify the source file.

---

64. N. Borenstein and N. Freed, "MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies," *IETF*, June 1992, http://www.ietf.org/rfc/rfc1341.txt.

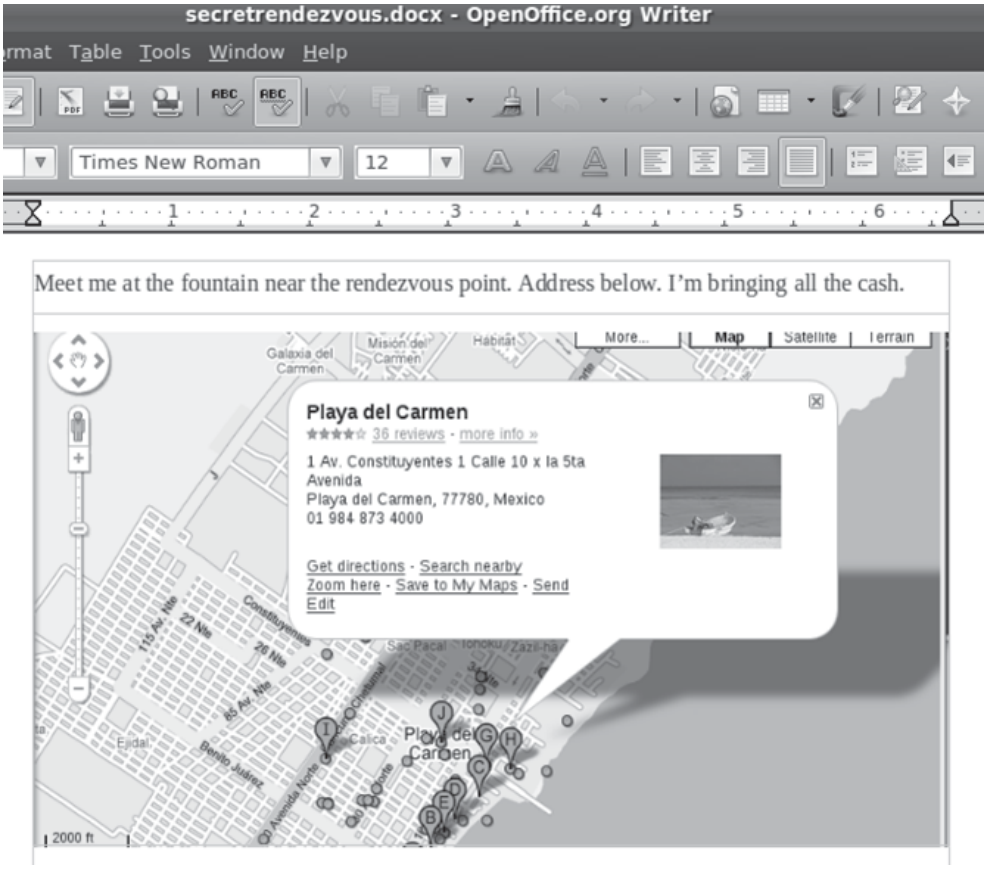


Figure 4–41. The attachment that Ann sent to "mistersecretx@aol.com," opened using OpenOffice. (Image ©2009 Google—Map data ©2009 INEGI.)

In this case, we open the attachment using OpenOffice, as shown in Figure 4–41. At the top of the document is the text, "Meet me at the fountain near the rendezvous point. Address below. I'm bringing all the cash." This is followed by an image of what appears to be a map with an address:

```
Playa del Carmen
1 Av Constituyentes 1 Calle 10 x la 5ta
Avenida
Playa del Carmen, 77780, Mexico
01 984 873 4000
```

Taking things one step further, we can carve out the image file embedded in the document. This can be useful for correlating with evidence from other sources, such as hard drive analysis. Carving embedded images is easy with .docx files. First, let's unzip the file: