Global Information Assurance Certification Paper

Forensic analysis of a provided image

GFCA practical version 2.0

Option 1

Rudolph Pereira

26 March 2005

Table of Contents

Notational conventions

Throughout this document, the following notational conventions will be used:

```
commands will appear like this
output (e.g. command output) will appear like this
```

0 Abstract

The following report outlines work done as part of the practical aspect of the GIAC Certified Forensics Analyst certification program. In it, the analysis of a supplied image is detailed. The analysis is meant to demonstrate the application of theory and skills taught in the SANS Track 8 Forensics course, as well as other applicable Computer/IT forensics knowledge.

The situation presented as background to the case is meant to replicate a real-life scenario: a typical harassment allegation in which IT/Communications resources are involved to only a small extent, at least at first.

As detailed in the report, the analysis eventually finds strong evidence that supports the allegations and that also indicates the offender committed a criminal act - interception of communications. Details of the specific statutes, issues surrounding them, and the possible punishments that could be levied against the offender are also covered.

Finally, the report presents some recommendations on both additional steps that might be taken as part of the investigation, and general issues that might need to be resolved to reduce or eliminate similar incidents in future.

1 Executive Summary

A sales representative employed by CC Terminals, Leila Conlay, has alleged that another employee, Robert Lawrence, also a sales representative, has harassed her. As part of the investigation into the allegation, Robert Lawrence's cubicle was searched and a USB flashdrive (i.e. a portable storage device) was found, which was examined using computer forensics techniques. In the course of that examination, three emails were found that appear to have been sent from Mr Lawrence to Ms Conlay over the course of four days. Additional files were also recovered from the storage device, again using forensically-sound techniques, that indicate Mr Lawrence obtained and used software that allowed him to intercept Ms Conlay's private communications in contravention of telecommunications interceptions statutes. Included in this intercepted traffic was an email sent by Ms Conlay in which she organises a social meeting with a third party.
Arranging the information obtained from the storage device into chronological order, it can clearly be seen that subsequent to Mr Lawrence expressing a romantic interest in Ms Conlay and Ms Conlay rejecting his advances, Mr Lawrence became more aggressive and intent on pursing Ms Conlay. He used the interception software to find out where Ms Conlay would be on the evening of 28 October so that he could orchestrate a run-in with Ms Conlay that appeared to her to be a coincidence. The final email suggests Mr Lawrence then attempted to threaten or otherwise convince Ms Conlay into rethinking her rejection of him.
In summary, the forensic analysis, detailed in the rest of this document has found evidence suggesting that Mr Lawrence did harass Ms Conlay and also that he has broken one or more statutes relating to telecommunication interception during his actions.

2 Analysis

In this section, we present details of the forensic analysis done on the supplied image. Section 2.1 lists the forensic tools used in the analysis and Sections 2.2 through 2.4 (inclusive) outline, in chronological order, the forensic analysis performed.

2.1 Forensic Analysis Tools/Resources

2.1.1 Forensic Workstation

The base forensic analysis workstation was a Compaq nc6000 laptop running Debian GNU/Linux (Sarge/Release 3.1)[1]. No binary/program files from the image provided were ever executed. In addition, all analysis was performed using a non-privileged user to ensure any inadvertent compromise would not damage the forensics environment completely.

2.1.2 Forensic analysis software/tools

The majority of the forensic analysis was performed using tools from Brian Carrier's sleuthkit, as well as a number of standard UNIX/Linux utilities. For completeness, the packages and versions for the tools used are listed; note that although these are Debian-customised versions, these customisations are not forensically relevant and can be ignored for our purposes.

Package: sleuthkit
Version: 1.73-6
Tools used: mmls fls mactime istat ifind icat fsstat

Package: coreutils
Version: 5.2.1-2
Tools used: dd

Package: file
Version: 4.12-1
Tools used: file

Package: binutils
Version: 2.15-5
Tools used: strings

Package: dpkg
Version: 1.10.27
Tools used: md5sum

Package: gzip
Version: 1.3.5-9
Tools used: gunzip

Package: ethereal
Version: 0.10.9-1
Tools used: ethereal

1 As the analysis was performed on a Linux system using command-line tools, the command output of tools shown throughout this document is equivalent to “screen shots” of the output of those tools.

Package: wv
Version: 1.0.2-0.1
Tools used: wvSummary

Note: unless otherwise specified, it can be assumed that commands were run and analysis done on the base Linux installation.

2.2 Details of supplied evidence

As part of the investigation, an image (<http://www.giac.org/GCFAPractical2.0-USBImageAndInfo.zip.gz>) has been provided which will be analysed as detailed in the rest of this document. The chain of custody form for the image has the following information:

- Tag #: USBFD-64531026-RL-001
- Description: 64M Lexar Media JumpDrive
- Serial #: JDSP064-04-5000C
- Image: USBFD-64531026-RL-001.img
- MD5: 338ecf17b7fc85bbb2d5ae2bbc729dd5

2.3 Forensic analysis of image

2.3.1 Initial Analysis (the "10000-foot view")

2.3.1.1 Verification of evidence

Initially, a quick analysis was done of the image to get an idea of the magnitude and type of work required for analysis, and to be able to formulate a plan of attack.

Firstly, the command **`file`**, which looks at a database of signatures to categorise the type of file given, was run on the downloaded image:

```
$ file GCFAPractical2.0-USBImageAndInfo.zip.gz
GCFAPractical2.0-USBImageAndInfo.zip.gz: gzip compressed data, was "USBFD-
64531026-RL-001.img", from Unix, max compression
```

So far, so good - the custody form listed the image name as "USBFD-64531026-RL-001.img". Next, given that **`file`** indicated it was a gzip-compressed file, the command **`gunzip`**, which uncompresses files previously compressed using Lempel-Ziv encoding, was used to obtain the original uncompressed image data - that data was written to the file “USBFD-64531026-RL-001.img”.
To confirm that indeed we were looking at a copy of the original file and hence, evidence that was forensically sound, the **`md5sum`** command, which outputs the 128-bit fixed-length MD5[2] hash of a file, was run:

```
$ md5sum USBFD-64531026-RL-001.img
338ecf17b7fc85bbb2d5ae2bbc729dd5  USBFD-64531026-RL-001.img
```

which we can see outputs an MD5 hash that agrees with the custody form, hence we can say that the contents of both files/images/devices are the same.

2 This can be likened to a unique digital fingerprint for the file. The MD5 hash function has the properties that the output of the function cannot be used to derive the input, and that it is mathematically improbable that two files of differing content will have the same hash value.

2.3.1.2 Storage basics

Before proceeding, it is necessary to have a basic understanding of how disks/storage devices, partitions, filesystems and files/directories fit together.

In general, higher layers in the hierarchy present more abstract or alternative views of the information provided at lower layers. At the highest layer, the *filename* layer, we see the typical representation of our useful information as files, often grouped into directories. Working in conjunction with that layer is the next layer down, the *metadata* layer, which stores the location of the file/directory data (i.e. the data blocks holding file/directory data) as well as time, ownership, and other data-about-the-data inside the next layer down, the *filesystem* layer.
The filesystem is responsible for presenting the blocks or sectors of the next layer down, the *media management* layer, to the metadata layer so that they can be allocated to hold our information (in files/directories).
The primary use of the *media management* layer is to subdivide the lowest layer, the *disk* layer, into partitions, which can be defined most simply as contiguous sections of the disk within which higher layers operate. Partitions are necessary so that multiple operating systems may reside on and manage, according to their filesystem, their own areas of the disk layer.
Finally, at the most fundamental level of the hierarchy - the *disk* layer - we have the storage device itself, such as the USB flashdrive we are examining here, storing data using whatever physical process the device uses, such as in non-volatile RAM (in a USB device) or the orientation of magnetic particles (in a hard disk). This layer is simply presented as a contiguous range of sectors or blocks ranging from zero to the size of the storage device to the media management layer.

2.3.1.3 Extracting relevant data for analysis

Although we know the image likely came from a PC, **`file`** was again run on the extracted image just to make sure:

```
$ file USBFD-64531026-RL-001.img
USBFD-64531026-RL-001.img: x86 boot sector
```

This indicated it did indeed come from an (x86) PC, and hence we can be sure it uses the usual partitioning format for IBM-compatible PCs, the DOS partition-table format/scheme.

Next, **`mmls`**,which displays an image's layout at the media management layer (i.e. partition layer) was used to display the partitions of the image:

```
$ mmls -t dos USBFD-64531026-RL-001.img
DOS Partition Table
Units are in 512-byte sectors

     Slot    Start        End          Length       Description
00:  -----   0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----   0000000001   0000000031   0000000031   Unallocated
02:  00:00   0000000032   0000121950   0000121919   DOS FAT16 (0x04)
```

From the **`mmls`** output, we can see that the partition we are primarily interested in lies in the sector range [32,121919]. The other "partitions" - more properly thought of as artefacts of how **`mmls`** looks at the disk/image, are:

0: the primary table - what we're looking at (via **`mmls`**)
1: an unused part of the disk

Just to make sure there isn't anything interesting in either the primary table or unallocated space, those sectors were also extracted with the **dd** command, which can copy ranges of data from one file to another, using the partition information supplied by **mmls** above. Then, the command **strings**, which displays sequences of printable characters, was run against the extracted data; this turned up nothing out of the ordinary. Together with the description we are given of the alleged offender:

> Robert Lawrence is employed at CC Terminals, a credit card processing firm. Robert works as a sales representative, selling credit card processing terminals.

- from which we can surmise the alleged offender very likely lacked in-depth knowledge of hiding data or subverting forensic methods - and later forensic analysis, we can be fairly sure that we're not missing anything so far by ignoring these parts of the image.
To get a copy of the areas we'd like to examine closely, we dump the partition at slot 0, using the values from **mmls** above passed as parameters to **dd**, to a file for further analysis:

```
$ dd if=USBFD-64531026-RL-001.img of=part1.img bs=512 count=121919 skip=32
```

"part1.img" now contains the data from the only “real” partition on the complete image.
To ensure our analysis does not alter the evidence while we are working on it, we get an MD5 hash of the image using **md5sum** and keep it for later comparison:

```
$ md5sum part1.img
5f830a763e2144483f78113a8844ad52  part1.img
```

To find out what type of partition we are dealing with, **file** is again used:

```
$ file part1.img
part1.img: x86 boot sector, code offset 0x3c, OEM-ID "MSWIN4.1",
sectors/cluster 2, root entries 512, Media descriptor 0xf8,
sectors/FAT 239, heads 17, hidden sectors 32, sectors 121919 (volumes
> 32 MB) , serial number 0x0, unlabelled, FAT (16 bit)
```

This indicates we are dealing with a FAT16 partition, and can adjust command invocation where necessary.
Note that we will refer to the file "part1.img" as the image, unless otherwise noted, for the rest of this document.

2.3.1.4 A quick look at the image

Linux natively provides the useful ability to "loopback mount" a filesystem image so that it looks like a normal partition (or disk). In our case, this was used to see what files were visible in the partition before any recovery or forensic analysis steps were taken: this can be likened to what a normal user would see were they to plug the drive into another machine. The mount command was also supplied with a number of options to disallow writing, execution, privilege changes or device files to be recognised by the OS, going some way to making sure the mounted image/files could not affect the analysis workstation. Hence:

```
# mount -t msdos -o ro,noexec,nodev,nosuid,loop part1.img mnt
# ls -l mnt/
total 60
-rwxr--r--  1 root root 19968 2004-10-28 18:24 coffee.doc
-rwxr--r--  1 root root 19968 2004-10-25 07:32 her.doc
-rwxr--r--  1 root root 19968 2004-10-26 07:48 hey.doc
```

We'll examine the relevant bits of the metadata shown above later, but for now, we'll copy out and note the MD5 hashes of (what appear to be) the MS Word documents, again, for later analysis:

```
# cp mnt/* copied-files/*
# md5sum mnt/*
a833c58689596eda15a27c931e0c76d1  mnt/coffee.doc
9785a777c5286738f9deb73d8bc57978  mnt/her.doc
ca601d4f8138717dca4de07a8ec19ed1  mnt/hey.doc
```

And just to confirm that they really are MS Word documents as their extensions suggest:

```
$ file copied-files/*.doc
copied-files/coffee.doc: Microsoft Office Document
copied-files/her.doc:    Microsoft Office Document
copied-files/hey.doc:    Microsoft Office Document
```

Given this view of the filesystem doesn't appear to offer any more information, we'll unmount the image and proceed down another track.

`fls`, which lists files and directories, including deleted files for which enough information still remains, was then run on the image, providing a slightly different view of things:

```
$ fls -f fat16 part1.img
r/r 3:  her.doc
r/r 4:  hey.doc
r/r * 7:        WinPcap_3_1_beta_3.exe (_INPCA~1.EXE)
r/r * 10:       WinPcap_3_1_beta_3.exe (_INPCA~1.EXE)
r/r * 12:       WinDump.exe (_INDUMP.EXE)
r/r * 14:       WinDump.exe (_INDUMP.EXE)
r/r * 15:       _apture
r/r * 16:       _ap.gif
r/r * 17:       _ap.gif
r/r 18: coffee.doc
```

Apart from the three MS Word documents copied above, we also see a number of deleted files, which we'll extract and examine later. Lastly, before delving into a detailed forensic analysis, we can use the metadata (specifically, data related to file activity) available to construct a timeline of file activity. Most filesystems, including the FAT16 filesystem being analysed here, store three timestamps for all files on the filesystem: the last modified (M) time, last accessed (A) time, and the last time the inode[3] directory entry for the file was changed (C). This is commonly referred to as the MAC (modified, accessed, changed) times or information. Constructing a MAC timeline gives us a chronological framework within which we can focus on particular events. To get an initial timeline, we can use **`mactime`**, which outputs a text representation of file activity based on the output of other tools (e.g. **`fls`**), as follows:

```
$ fls -f fat16 -m / part1.img | mactime -b - > timeline.mactime.out
```

The contents of the file "timeline.mactime.out", without annotations, appears in appendix A.

Next, **`strings`** was run over the whole image in case we quickly wanted to see if a particular string was present; the option to print the string's location was also passed to

3 UNIX-like filesystems refer to directory entries as index-nodes or inode; though strictly speaking inode is a UNIX filesystem term, inode and directory entry will be used interchangeably throughout this document

strings so that we could see it's location and possibly focus on data areas around it:

```
$ strings -t d part1.img > part1.img.strings
```

Finally, **fsstat** was run to display the sector size for the filesystem - which was found to be 512. This indicates that each data block (at least as far as the sleuthkit tools are concerned) allocated to a file or directory is 512 bytes in length; files must hence take up an integral number of 512-byte blocks on the storage device.

2.3.1.5 Dirty-word list

As with all forensics investigations, at this point, a “dirty word list” was created, the intention being that key words or phrases can be entered into the list as they are found and that list then used to seed new searches or find new “leads”. Entries were added to this list during the course of the investigation, and the final list can be seen in Appendix C.

2.3.2 In-depth analysis

2.3.2.1 Introduction

Given the output of **fls** above, we have a rough idea that there are a number of MS Word documents, which we've already got copies of, and a few other files that have been deleted but that we may be able to recover. We also have a timeline, not yet analysed, that we can use to guide our analysis. Before we do that though, we need to understand some of the properties of the FAT16 filesystem so that we can proceed with a thorough analysis

2.3.2.2 Filesystem properties and eccentricities

2.3.2.2.1 FAT16 filesystem properties

To better understand the information the timeline output/analysis gives us, we need to be aware of certain properties of the FAT16 filesystem and how the utilities we are using for analysis - primarily **mactime** - display filesystem information given these properties. Namely:

- MAC times are saved in localtime. This is not relevant apart from knowing that timezone offsets do not need to be considered
- the access time is only recorded accurate to the day[4] (24 hours), if at all[5]; **mactime** thus shows all access times as being at midnight on the day the file was accessed
- the seconds part of the modification time is stored accurate to the nearest two seconds[6]
- the creation time of the file, rather than the last time the file's inode/directory entry was changed, is stored[7]
- FAT16 has no native concept of permissions or access-control lists/entries, thus, **mactime** will display the equivalent to "full" permissions, or in UNIX terms, "rwx" (readable, writeable, executable) for the owner of the file, group-owner of the file, and everyone else

4 See section titled “General Notes on time” in <http://www.sleuthkit.org/sleuthkit/docs/skins_fat.html> [SLKTDOC], or [FATSPEC00] pp 24.
5 See [FATSPEC00] pp 27.
6 See [FATSPEC00] pp 24.
7 See [FATSPEC00] pp 24.

- as with permissions, there is no concept of an owner or a group-owner, thus **mactime** displays these as having a user-id and group-id of 0, equivalent to root on UNIX-like filesystems.
- the FAT16 filesystem does not store the time of last use of the filesystem
- long filenames, for example those exceeding the normal MS-DOS 8-character name, 3 character suffix, are stored in the filesystem as a series of directory entries, only one of which will point to real data. Since the sleuthkit utilities used deal with this transparently, it is not necessary to deal with this characteristic on it's own other than to understand that some directory entries are used to store fragments of long filenames
- deleted files are marked deleted by writing the (hex) value 0xE5 to the first character of the MS-DOS 8.3 filename. As 0xE5 is the hex code for the ASCII underscore ("_"), files that have been deleted will be displayed (by utilities such as **fls** that display such information) as stored, i.e. with their first character replaced by an underscore. In cases where the file has a long-file component, that component will remain intact, and can be used to work out the complete, original 8.3 filename if necessary
- although FAT16 uses the concept of a cluster - a number of sectors - as the base storage unit, the sleuthkit tools ignore this and display/calculate everything in terms of sectors. We will use the terms sectors and blocks interchangeably, and it can be assumed that a sector and a block are the same size

2.3.2.2.2 Double file entries

If we look at the output of **fls**:

```
$ fls -f fat16 part1.img
r/r 3:  her.doc
r/r 4:  hey.doc
r/r * 7:        WinPcap_3_1_beta_3.exe (_INPCA~1.EXE)
r/r * 10:       WinPcap_3_1_beta_3.exe (_INPCA~1.EXE)
r/r * 12:       WinDump.exe (_INDUMP.EXE)
r/r * 14:       WinDump.exe (_INDUMP.EXE)
r/r * 15:       _apture
r/r * 16:       _ap.gif
r/r * 17:       _ap.gif
r/r 18: coffee.doc
```

more closely, we see a number of "double entries". For example, for the file "WinPcap_3_1_beta_3.exe", we see directory/inode entries 7 and 10 present. **istat** shows:

```
$ istat -f fat16 part1.img 7
Directory Entry: 7
Not Allocated
File Attributes: File, Archive
Size: 0
Name: _INPCA~1.EXE

Directory Entry Times:
Written:        Wed Oct 27 16:23:56 2004
Accessed:       Wed Oct 27 00:00:00 2004
Created:        Wed Oct 27 16:23:54 2004

Sectors:

Recovery:
File recovery not possible

$ istat -f fat16 part1.img 10
```

```
Directory Entry: 10
Not Allocated
File Attributes: File, Archive
Size: 485810
Name: _INPCA~1.EXE

Directory Entry Times:
Written:        Wed Oct 27 16:23:50 2004
Accessed:       Thu Oct 28 00:00:00 2004
Created:        Wed Oct 27 16:23:54 2004

Sectors:
591 592 593 594 595 596 597 598
...
```

Similar entries are present for the files "WinDump.exe" and "_ap.gif". As the **`istat`** output shows, the entries have create/write times that are approximately the same, but one of the entries has a zero file size; in addition, the attributes do not indicate the entries are anything to do with long-filename support.
Eventually, this was tracked down to an unusual characteristic of applications running under recent Windows operating systems (Windows 2000 and later): when saving files from within an application (e.g. using the “Save As” option) an empty file with the requested filename is initially created and then deleted, after which another file with the same filename but different directory entry/inode is created and the file data stored in this second file.
Please see appendix B for more details of the above behaviour.

2.3.2.3 Timeline analysis

Keeping in mind that access time is only recorded accurate to the day, that the seconds part of the modification time is stored accurate to the nearest two seconds and that the creation time of the file, rather than the last time the file's inode/directory entry was changed, is stored, we can now start to analyse the timeline obtained above. For clarity, the lines of the timeline below have been numbered and the columns below are as outputted by mactime with the line number prepended, i.e. (in order):

line-number|date|time|file-size|(m)odified/(a)ccessed/(c)hanged|permissions|owner|group|inode number|filename

```
1 Mon Oct 25 2004 00:00:00    19968 .a. -/-rwxrwxrwx 0        0        3
/her.doc
2 Mon Oct 25 2004 08:32:06    19968 ..c -/-rwxrwxrwx 0        0        3
/her.doc
3 Mon Oct 25 2004 08:32:08    19968 m.. -/-rwxrwxrwx 0        0        3
/her.doc
```

Lines 1-3 indicate the file "her.doc" was created and saved at 08:32 without later modification. Line 1 indicates the file was accessed on Oct 25, line two indicates the file was created at 8:32:06, and line 3 shows it being modified (up to) two seconds later.

```
4 Tue Oct 26 2004 00:00:00    19968 .a. -/-rwxrwxrwx 0        0        4
/hey.doc
5 Tue Oct 26 2004 08:48:06    19968 ..c -/-rwxrwxrwx 0        0        4
/hey.doc
6 Tue Oct 26 2004 08:48:10    19968 m.. -/-rwxrwxrwx 0        0        4
/hey.doc
```

Lines 4-6 are similar to lines 1-3; again, we see a file created and shortly after, modified.

This pattern will likely show up with one-off files/documents that are typed up, saved and never modified or accessed later.

```
      7 Wed Oct 27 2004 00:00:00   485810 .a. -/-rwxrwxrwx 0        0        7
/WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
      8                            450560 .a. -/-rwxrwxrwx 0        0        12
/WinDump.exe (_INDUMP.EXE) (deleted)
      9 Wed Oct 27 2004 16:23:50   485810 m.. -/-rwxrwxrwx 0        0        10
/WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
      10 Wed Oct 27 2004 16:23:54   485810 ..c -/-rwxrwxrwx 0        0        10
/WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
      11                            485810 ..c -/-rwxrwxrwx 0        0        7
/WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
      12 Wed Oct 27 2004 16:23:56   485810 m.. -/-rwxrwxrwx 0        0        7
/WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
```

Lines 7 and 9-12 relate to a file - "WinPcap_3_1_beta_3.exe" - that was saved to the filesystem. Two directory entries (with directory/inode numbers 7 and 10) are listed due to the double-entry behaviour discussed above. **`mactime`** also displays the files as being deleted, though as the FAT16 filesystem does not record deletion in any metadata, we cannot tell exactly when the deletion occurred.

```
      13 Wed Oct 27 2004 16:24:02   450560 m.. -/-rwxrwxrwx 0        0        14
/WinDump.exe (_INDUMP.EXE) (deleted)
      14 Wed Oct 27 2004 16:24:04   450560 ..c -/-rwxrwxrwx 0        0        14
/WinDump.exe (_INDUMP.EXE) (deleted)
      15                            450560 ..c -/-rwxrwxrwx 0        0        12
/WinDump.exe (_INDUMP.EXE) (deleted)
      16 Wed Oct 27 2004 16:24:06   450560 m.. -/-rwxrwxrwx 0        0        12
/WinDump.exe (_INDUMP.EXE) (deleted)
```

Again, lines 13-16 show another file - "WinDump.exe" saved to the filesystem. It too has been deleted (at an unknown time).

```
      17 Thu Oct 28 2004 00:00:00    19968 .a. -/-rwxrwxrwx 0        0        18
/coffee.doc
      18                              8814 .a. -/-rwxrwxrwx 0        0        17
/_ap.gif (deleted)
      19                             53056 .a. -/-rwxrwxrwx 0        0        15
/_apture (deleted)
      20                            450560 .a. -/-rwxrwxrwx 0        0        14
/WinDump.exe (_INDUMP.EXE) (deleted)
      21                            485810 .a. -/-rwxrwxrwx 0        0        10
/WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
      22                              8814 .a. -/-rwxrwxrwx 0        0        16
/_ap.gif (deleted)
```

Lines 17-22 give a list of the files that were accessed on Oct 28 (some time during that day). Most significantly, we can now see that "WinDump.exe" and "WinPcap_3_1_beta_3.exe" were accessed during the day and hence still existed on the filesystem - that is, they had not been deleted.

```
      23 Thu Oct 28 2004 11:08:24    53056 ..c -/-rwxrwxrwx 0        0        15
/_apture (deleted)
      24 Thu Oct 28 2004 11:11:00    53056 m.. -/-rwxrwxrwx 0        0        15
/_apture (deleted)
      25 Thu Oct 28 2004 11:17:44     8814 ..c -/-rwxrwxrwx 0        0        16
```

```
/_ap.gif (deleted)
      26                              8814 ..c -/-rwxrwxrwx 0        0        17
/_ap.gif (deleted)
      27 Thu Oct 28 2004 11:17:46     8814 m.. -/-rwxrwxrwx 0        0        16
/_ap.gif (deleted)
      28                              8814 m.. -/-rwxrwxrwx 0        0        17
/_ap.gif (deleted)
      29 Thu Oct 28 2004 19:24:46    19968 ..c -/-rwxrwxrwx 0        0        18
/coffee.doc
      30 Thu Oct 28 2004 19:24:48    19968 m.. -/-rwxrwxrwx 0        0        18
/coffee.doc
```

Finally, lines 23-30 show a few more files being created, some of which were also deleted (time unknown).

2.3.2.4 Recovery of deleted files

Both **`fls`** and the **`mactime`**-generated timeline above indicate there were a number of files that were deleted from the filesystem that are of interest, namely: WinDump.exe, WinPcap_3_1_beta_3.exe, _ap.gif and _apture
Focussing on the **`fls`** output for just those files:

```
r/r * 7:        WinPcap_3_1_beta_3.exe (_INPCA~1.EXE)
r/r * 10:       WinPcap_3_1_beta_3.exe (_INPCA~1.EXE)
r/r * 12:       WinDump.exe (_INDUMP.EXE)
r/r * 14:       WinDump.exe (_INDUMP.EXE)
r/r * 15:       _apture
r/r * 16:       _ap.gif
r/r * 17:       _ap.gif
```

`istat` and **`icat`** (from the sleuthkit) can now be used to attempt recovery of the deleted files.
We start off with **`istat`** to get some more information about the directory entry for the file, particularly to find out whether the entry points to data blocks and whether recovery is possible. Taking "WinPcap_3_1_beta_3.exe":

```
$ istat -f fat16 part1.img 7
Directory Entry: 7
Not Allocated
File Attributes: File, Archive
Size: 0
Name: _INPCA~1.EXE
...
```

the filesize indicates this was not the inode we were looking for, whereas looking at the other entry for this file (directory entry/inode 10, as listed by **`fls`** above):

```
$ istat -f fat16 part1.img 10
Directory Entry: 10
Not Allocated
File Attributes: File, Archive
Size: 485810
Name: _INPCA~1.EXE

Directory Entry Times:
Written:        Wed Oct 27 16:23:50 2004
Accessed:       Thu Oct 28 00:00:00 2004
Created:        Wed Oct 27 16:23:54 2004
```

```
Sectors:
591 592 593 594 595 596 597 598
599 600 601 602 603 604 605 606
607 608 609 610 611 612 613 614
615 616 617 618 619 620 621 622
623 624 625 626 627 628 629 630

Recovery:
File recovery not possible
```

Unfortunately, we cannot recover this file using **icat**: most likely this is because it's data blocks have been used by another file. To confirm this, **ifind** can be used to search for the inode using a given datablock number, which in this case shows (using one of the data blocks listed in the inode above):

```
$ ifind -f fat part1.img -d 591
18
```

which is being used by the file "coffee.doc" according to the **fls** output above. The reason the inode for “WinPcap_3_1_beta_3.exe” points to the data blocks of another file, “coffee.doc”, is that the FAT filesystem stores a pointer to the start of the datablock “chain” in the directory entry and then relies on the entries in the file allocation table (FAT) to link from one to the next. In the case of deleted entries - as we are seeing here - the directory entry for the file is marked deleted by overwriting the first character of the filename, but all other information, including the pointer to the chain of entries in the FAT, remain untouched. Should another file - in this case, “coffee.doc” - be allocated to one or more of those freed blocks/FAT entries, the deleted file will point to the chain for the new file rather than the old. Hence, what **icat** is showing us is the group of blocks associated with "coffee.doc" rather than the (deleted) file "WinPcap_3_1_beta_3.exe".

The discrepancy between the file size and the number of blocks also supports this:

```
$ istat -f fat16 part1.img 10
...
Size: 485810
...
Sectors:
591 592 593 594 595 596 597 598
599 600 601 602 603 604 605 606
607 608 609 610 611 612 613 614
615 616 617 618 619 620 621 622
623 624 625 626 627 628 629 630

Recovery:
...
```

We should be seeing *size/sector-size* sectors (i.e. 485810/512 = 949 [integral sectors]) listed rather than the 40 blocks listed here. Recalling that **ifind** (above) told us inode 18 was allocated sector 591, let's look at the **istat** output for that inode:

```
$ istat -f fat16 part1.img 18
Directory Entry: 18
Allocated
File Attributes: File, Archive
Size: 19968
Name: coffee.doc

Directory Entry Times:
Written:        Thu Oct 28 19:24:48 2004
```

```
Accessed:       Thu Oct 28 00:00:00 2004
Created:        Thu Oct 28 19:24:46 2004

Sectors:
591 592 593 594 595 596 597 598
599 600 601 602 603 604 605 606
607 608 609 610 611 612 613 614
615 616 617 618 619 620 621 622
623 624 625 626 627 628 629 630
```

As we can see, the sector list is exactly the same as the one shown for "WinPcap_3_1_beta_3.exe".

From the above, we now know that the first 40 sectors (sectors 591-630) from the file we are trying recover (WinPcap_3_1_beta_3.exe) have been reused/overwritten by the data in “coffee.doc”. What about the rest of the file, namely the other 949-40 = 919 sectors worth of data?

Before answering the above question, we need to cover another tidbit of information related to filesystems: most filesystems, FAT included, attempt to reduce fragmentation (which slows down I/O) by keeping files unfragmented and contiguous. It is likely that the data for WinPcap_3_1_beta_3.exe was originally saved to the contiguous range of sectors 591 to 1540 (591+949) and although the first 40 sectors have since been overwritten, if the others have not they are likely to contain the rest of the data for that file.

Looking at the timeline and inode entries, we can indeed say that the only files related to this region are "coffee.doc" and "WinPcap_3_1_beta_3.exe", so we can now proceed with attempting to recover the remaining part of the file "WinPcap_3_1_beta_3.exe" from the image.

From the inode information above, we know that the remaining data for "WinPcap_3_1_beta_3.exe" likely resided in sectors 631 to 1540.

To take into account the position of the remaining file data on the image, and the amount of data we need to read, we perform a couple of calculations:

- the number of bytes to skip (seek) into the partition/image is given by:
  *skip = (sector size) * (number of sectors to start of file-fragment)*
  *= 512 * 631*
  = 323072

- the number of bytes to read from the above position is given by:
  *count = (size of "WinPcap_3_1_beta_3.exe") - ((size of "coffee.doc" in sectors) * (sector size))*
  *= 485810 - (40 * 512)*
  = 465330

We can now use **dd** to read the blocks and write them to another file; and we also take the MD5 hash for later reference:

```
$ dd if=part1.img bs=1 skip=323072 count=465330 of=undel-
files/WinPcap_3_1_beta_3.exe.partial.fromimg
$ md5sum undel-files/WinPcap_3_1_beta_3.exe.partial.fromimg
70553e1c186284f46b46c9521bba629b undel-
files/WinPcap_3_1_beta_3.exe.partial.fromimg
```

Now that we've done the tricky recovery, we can recover the other deleted files using the combination of **istat** to find the inode with pointing to data blocks and **icat -r**.
The filenames in question, their "real" (i.e. pointing to data blocks) inode and the associated data block ranges are (from **istat**):

WinDump.exe: inode 14, sectors 1541-2420
_apture: inode 15, sectors 2421-2524
_ap.gif: inode 17, sectors 2525-2542

We can now recover the above three files using **icat -r**; for reference later the md5sums are:

```
$ md5sum undel-files/*
40f95f34699273fd4d681bd68a7a3ab5  undel-files/_ap.gif
f7db6e0ecb4c3c51218d918c9c711cbd  undel-files/_apture
79375b77975aa53a1b0507496107bff7  undel-files/WinDump.exe
```

Just for completeness, we also run **file** over the three most recently recovered files:

```
$ file undel-files/_ap.gif undel-files/_apture undel-files/WinDump.exe
undel-files/_ap.gif:     GIF image data, version 89a, 300 x 200
undel-files/_apture:     tcpdump capture file (little-endian) - version
2.4 (Ethernet, capture length 4096)
undel-files/WinDump.exe: MS-DOS executable (EXE), OS/2 or MS Windows
```

2.3.2.5 Slack space

Slack space is the space in the last sector (or cluster, or allocation unit) allocated to a file but not occupied by the file's data. It is a result of block allocation being in whole numbers of allocation units; for example: a file that occupied 1.5 sectors worth of data would have two sectors allocated to it, with only half the last sector actually containing data. The reason slack space is relevant at this point is that it may be used to hide data.

Our analysis so far has not seen any evidence suggesting that data-hiding methods (such as are used in manipulating slack space) have been used, and in addition, we know that the alleged offender - Mr Lawrence - was unlikely to have enough knowledge about filesystem internals to hide data in slack space.
Nevertheless, we examine slack space for the recovered files for completeness. We do this by using the following algorithm: for each file (directory entry), calculate how much of the last sector would be free (i.e. the slack space), then examine anything in that data area.
As expected, no information relevant to the case was found.

2.3.2.6 Summary of recovered files

In the above sections, we have managed to recover complete copies of all files bar one, and in that case we recovered the last ~95% of it. To recap, the filenames, sizes and MD5 hashes for the recovered files are:

```
 coffee.doc 20480 b3e93d38943d122cae87c38134be7a22
    her.doc 20480 62069575228d762000fd7303d3cfda61
    hey.doc 20480 9d072019092340f171a0b1f99ce036d8
    _ap.gif 9216 40f95f34699273fd4d681bd68a7a3ab5
    _apture 53248 f7db6e0ecb4c3c51218d918c9c711cbd
WinDump.exe 450560 79375b77975aa53a1b0507496107bff7
```

```
WinPcap_3_1_beta_3.exe.partial.fromimg 465330 70553e1c186284f46b46c9521bba629b
```

[Note that the true name of the last file is actually "WinPcap_3_1_beta_3.exe"; the name used indicates a partially recovered file to avoid confusion]

Unfortunately, we were not able to recover the following:

- complete filenames for files with only the short 8.3 MS-DOS filename (e.g. _ap.gif and _apture)
- file owner/group (as FAT16 does not know about these)

2.4 File content analysis

We'll now look a little deeper into the contents of the recovered files themselves

2.4.1 Document/Text files

According to **`file`**, the files coffee.doc, her.doc and hey.doc appear to be MS Word document files.

Using **`strings`**, we can quickly display the text contained in the files: (Note: some irrelevant strings omitted)

```
$ strings coffee.doc
...
Hey what gives?  I was drinking a coffee on thursday and saw you stop
buy with some guy!  You said you didn't want coffee with me, but
you'll go have it with some random guy???  He looked like a loser!
Guys like that are nothing but trouble.  I can't believe you did this
to me!  You should stick to your word, if you're not interested in
going to coffee with me then you shouldn't be going with anyone!  I
heard rumors about a "bad batch" of coffee, hope you don't get any...
Hey what gives
Robert Lawrence
Normal.dot
Robert Lawrence
Microsoft Word 10.0
Hey what gives
...

$ strings her.doc
...
Hey I saw you the other day.  I tried to say "hi", but you
disappeared???  That was a nice blue dress you were wearing.  I heard
that your car was giving you some trouble.  Maybe I can give you a
ride to work sometime, or maybe we can get dinner sometime?
Have a nice day
Hey I saw you the other day
Robert Lawrence
Normal.dot
Robert Lawrence
Microsoft Word 10.0
...

$ strings hey.doc
...
Hey!  Why are you being so mean?  I was just offering to help you out
with your car!  Don't tell me to get lost!  You should give me a
chance.  I'm a nice guy just trying to help you out, just because I
```

```
think you're cute doesn't mean I'm weird.  Perhaps coffee would be
better, when would be a good time for you?
Robert Lawrence
Normal.dot
Robert Lawrence
Microsoft Word 10.0
...
```

Apart from the actual text, the most interesting text strings are "Microsoft Word 10.0" and "Robert Lawrence". We can guess that these are metadata strings stored in the MS Word document file, and that they confirm the author as the suspect, Robert Lawrence, and that the application used to create the files was indeed MS Word (version 10.0). The useful utility **wvSummary**[8], which displays OLE file metadata, can be used to confirm this:

```
$ wvSummary coffee.doc
The title is Hey what gives
The subject is
The author is Robert Lawrence
The keywords are
The comments are
The template was Normal.dot
The last author was Robert Lawrence
The rev # was 1
The app name was Microsoft Word 10.0
...

$ wvSummary her.doc
...
The author is Robert Lawrence
...
The last author was Robert Lawrence
...

$ wvSummary hey.doc
...
The author is Robert Lawrence
...
The last author was Robert Lawrence
...
```

Hence we can say with much certainty that the suspect did author the documents, the text of which we see about (as outputted by **strings**).

2.4.2 Binary/executable files

Our file recovery activities above led to the recovery of two files that have file content (according to **strings**) or a filename indicating they are likely to be (PC/Windows) binary executables: WinDump.exe and WinPcap_3_1_beta_3.exe.partial.fromimg

Searching for the string "WinDump.exe" in the Google search engine leads us to the website <http://windump.polito.it/>, which states:

> WinDump is the porting to the Windows platform of tcpdump, the most used network sniffer/analyzer for UNIX. WinDump is fully compatible with tcpdump and can be used to watch and diagnose network traffic according to various complex rules. It can run under Windows 95/98/ME, and under Windows NT/2000/XP.

8 Part of the wv(ware) utilities, see <http://wvware.sourceforge.net/>

> WinDump uses a libpcap-compatible library for Windows, WinPcap, which is freely downloadable from the WinPcap site.

Hence we have quickly established what the executables we recovered are likely to be - part of a packet capture utility - and also, what they are likely to have been used for - (obviously) packet capture.

We do need to ensure though that the file content matches as well, rather than just the filenames. This is simply done, in theory: we can download copies of the pre-compiled binary files provided at the above (windump) site and compare[9] MD5 hashes. The only remaining hurdle is, of course, versions: different versions of the packet capture tools will have different sets of object code, hence different contents, and different MD5 hashes. We would be wasting our time downloading versions different to the ones we are analysing as their contents will be different.

For the file "WinPcap_3_1_beta_3.exe.partial.fromimg", the version is pretty obvious from the filename: version 3_1_beta_3. Digging around on the winpcap site, since the above version isn't the current/latest one mentioned in the main pages of the winpcap site, we find a link to a directory with that version[10] at <http://windump.polito.it/misc/bin/> (the windump site!) from which we download a copy.

Recalling that the recovery performed above for this file was a partial recovery, we have a bit more work to do before being able to do a straight comparison: for the downloaded file, we now have to obtain the latter section of the file comparable in size to the section of "WinPcap_3_1_beta_3.exe" we recovered. To do this, we make the following calculation:

- the number of bytes to skip (seek) into the partition is given by:

  *skip = (sector size) * (number of sectors used by "coffee.doc")*
  *= 512 * 40*
  = 20480

and use "dd" to copy the correct section of the file:

```
$ dd if=downloads/WinPcap_3_1_beta_3.exe bs=1 skip=20480 of=undel-
files/WinPcap_3_1_beta_3.exe.partial.fromdl
```

We can now analyse both file fragments:

```
$ md5sum WinPcap_3_1_beta_3.exe.partial.fromdl
WinPcap_3_1_beta_3.exe.partial.fromimg
70553e1c186284f46b46c9521bba629b WinPcap_3_1_beta_3.exe.partial.fromdl
70553e1c186284f46b46c9521bba629b WinPcap_3_1_beta_3.exe.partial.fromimg
```

This indicates that the data extracted from the image and that from the downloaded file are exactly the same. Although, again, we've only compared the last 95% of both files, given the structure of typical binary/program files, we can use the comparison and our previous analysis to say that the file "WinPcap_3_1_beta_3.exe" (winpcap version 3.1 beta 3) was almost certainly saved to and subsequently deleted from the partition/image.

In our analysis of the other binary/executable file, "WinDump.exe", we are faced with the problem that we cannot easily determine the version of the utility from the filename. Instead, we must look for "version strings" within the contents of the file itself. *Version strings* are simply strings in the executable code that have been inserted to

9 As stated previously, if the MD5 hashes of two files are the same, it is mathematically improbable that their contents are different. Hence comparing the MD5 hashes of two files for equality is virtually equivalent to comparing the contents of the files themselves.

10 listed under the "older versions" section on the page at <http://winpcap.polito.it/install/default.htm>

indicate the version of the program. Often they are part of resources that display this version number to the user via the program's interface, for example, in "about this program" dialogue boxes. To have a better idea of what kinds of versions we might be looking for, we visit the windump homepage again (<http://windump.polito.it/>) and see that the current version (as of this writing) is "3.8.3 beta". We can now use **strings** on the binary file, combined with **grep**, which extracts those strings matching a pattern given (in this case, we will search for any digit followed by a dot), to find those string patterns that match the windump versioning scheme:

```
$ strings WinDump.exe|egrep '[0-9\.]'
...
3.8.3 beta
3.8.3
0.8.1
3.1 beta2
...
```

The first of these suggests the version of WinDump may be "3.8.3 beta". As of this writing, this is the latest version of WinDump according to the windump homepage (above), so we download the file "WinDump.exe" (from <http://windump.polito.it/install/default.htm>) and compare the MD5 hashes of that file with the one extracted from the partition image:

```
$ md5sum downloads/WinDump.exe undel-files/WinDump.exe
79375b77975aa53a1b0507496107bff7  downloads/WinDump.exe
79375b77975aa53a1b0507496107bff7  undel-files/WinDump.exe
```

and again find that they match. Hence we can say that the contents of the files (and the version numbers we guessed above) are the same. There is no need to download and compile source code, as we have obtained binaries from a trusted source (the windump/winpcap sites) along with descriptions of their behaviour, and we know that the binaries on the analysed image are the same binaries as the ones we have (downloaded) from the windump/winpcap site.

Given the analysis of both binary/program files above, we can now say with certainty that the files required (WinDump.exe and WinPcap_3_1_beta_3.exe) for a working installation of a packet capture utility (WinDump) did exist (i.e. were saved) on the analysed image/partition.

2.4.3 Traffic capture file

According to **file**, the file "_apture" is:

```
$ file _apture
_apture: tcpdump capture file (little-endian) - version 2.4 (Ethernet,
capture length 4096)
```

Together with the knowledge that a packet capture utility was present on the system, we can guess that this file was saved from the output of the "WinDump" utility analysed above, and was likely named "capture".

As a quick first step, we can use the dependable **strings** to extract readable strings from the capture. Doing this, we see a number of what appear to be http requests, including the following:

```
$ strings _apture
...
```

```
curmbox=F000000001&HrsTest=&_HMaction=Send&FinalDest=&subaction=&plaintext
=&login=flowergirl96&msg=&start=&len=&attfile=&attlistfile=&eurl=&type=&sr
c=&ref=&ru=&msghdrid=b16479b18beec291196189c78555223c_1098692452&RTEbgcolo
r=&encodedto=SamGuarillo@hotmail.com&encodedcc=&encodedbcc=&deleteUponSend
=0&importance=&sigflag=&newmail=new&to=SamGuarillo@hotmail.com&cc=&bcc=&su
bject=RE%3A+coffee&body=Sure%2C+coffee+sounds+great.++Let%
27s+meet+at+the+coffee+shop+on+the+corner+Hollywood+and+McCadden.++It%
27s+a+nice+out+of+the+way+spot.%0D%0A%0D%0ASee+you+at+7pm%21%0D%0A%0D%0A-
Leila.6
```

which looks very much like the contents of a mail sent via a web-based mail provider.

We now turn to "ethereal", an X11-based packet capture/analysis utility, that can open and analyse previously-saved packet captures in many formats including the "tcpdump capture file" format being looked at here. Using ethereal, we can see the following information in the file "_apture":

- there are two network nodes with non-routable IP addresses (in the 192.168.2/24 reserved range) from which traffic was captured: 192.168.2.104 and 192.168.2.1
- The node with IP address 192.168.2.1 only appears to generate SNMP traps (in the capture); ethereal suggests it's MAC address (00:0c:41:50:29:2c) has a vendor ID of "LinksysG". It is likely this network node is a Linksys router (or access point) of some sort with a default configuration (why else would SNMP traps containing current TCP connections be broadcast?)
- The node with IP address 192.168.2.104 is the only one generating "real" network traffic, as in, it is making TCP connections to nodes outside the local network. This node has a MAC address of 00:90:4b:5e:e3:cf which ethereal says belongs to the vendor "GemtekTe"
- there are five remote TCP sources/destinations with IP/hostnames (at the time of this writing):

  Name: www.bay12.hotmail.com
  Address: 64.4.34.250

  Name: rad.msn.com
  Address: 207.68.178.16

  Name: h.msn.com
  Address: 207.68.177.124

  Name: unknown.Level3.net
  Address: 63.209.188.62

  Name: annyadvip2.doubleclick.net
  Address: 216.73.86.40

With that information in mind, we can start analysing the traffic to and from the node with IP address 192.168.2.104, in particular, we can try to find the packets containing the text shown above with **`strings`**. To do this, we can use the "follow TCP stream" capabilities of ethereal. Looking at packet 1[11] of the capture, we see that it is a connection initiated from 192.168.2.104 to 64.4.34.250; this is as good a place as any to start, so we "follow the TCP stream" from packet one and soon see that packet 5, in the same

11 Absolute packet numbers are used throughout

stream/connection, contains an HTTP POST. HTTP POSTs are used to send/upload data to a server, such as mail messages, and sure enough, we see the mail message with the text shown by **`strings`** (above) in packet 7.

We have now confirmed that the file "_apture" is a saved packet capture, likely generated by the "WinDump" utility from above, and that it contains captured traffic including a mail message sent from "Leila" (likely Leila Conway) to "SamGuarillo@hotmail.com" and discussing a location - "the coffee shop on the corner Hollywood and McCadden" - and a time at which they are to meet (7pm).

As to how the offender captured the traffic, we can only guess that either he was connected into the same hub that the target was, or, more likely these days, that the network being used was an 802.11 wireless LAN. The vendor IDs of "LinksysG" for the gateway, which suggests some kind of LinkSys-G access point, and "GemtekTe" which, according to their website at <http://www.gemtek.com.tw/> primarily sell wireless network devices for the target's network card - suggests the latter.

2.4.4 Image file

The remaining recovered file, "_ap.gif", is suggested by **`file`** as being a GIF image:

```
$ file _ap.gif
_ap.gif: GIF image data, version 89a, 300 x 200
```

Viewing the file, we see that it appears to be a saved image from the Microsoft "MapPoint" service (www.mappoint.com). In addition, there are indications that the location mentioned in the email (above) - "the corner Hollywood and McCadden" - have been highlighted on the saved map.

3 Connecting the threads

Using the results of the detailed analysis and the timeline above, we can now summarise the important points of the case

3.1 The timeline analysis, redux

Using the binary and contents analysis, we can connect the timeline's coarse chronological events with their intent or meaning. To make things a little clearer, rather than re-displaying the mactime output verbatim (for which the reader is referred to Appendix A), only the following data from the timeline will be presented:

```
line number, time/date, modified/access/changed flag, filename

 1 Mon Oct 25 2004 00:00:00    (access) /her.doc
 2 Mon Oct 25 2004 08:32:06    (change) /her.doc
 3 Mon Oct 25 2004 08:32:08    (modify) /her.doc
```

Given that the metadata of "her.doc", created above, lists the author as Robert Lawrence, we can say that he created the above document. In it, he expresses interest in the recipient, and suggests they go out on a date. The tone of the document suggests that it was sent as an email, and given Ms Conlay stated that Mr Lawrence had sent her emails, and also given the details of the rest of this case, including the information in the email sent by "Leila" in the capture below, we can say that it is likely this was one of the emails

sent to Ms Conlay by Mr Lawrence.

```
 4 Tue Oct 26 2004 00:00:00    (access) /hey.doc
 5 Tue Oct 26 2004 08:48:06    (change) /hey.doc
 6 Tue Oct 26 2004 08:48:10    (modify) /hey.doc
```

Here we see another document similar to the one above, created by Mr Lawrence, indicating his first proposal had been rejected (to his irritation) by Ms Conlay and suggesting another, perhaps more informal ("coffee") date. Again, from the tone, this was likely sent as an email to Ms Conlay.

```
 7 Wed Oct 27 2004 00:00:00    (access) /WinPcap_3_1_beta_3.exe
 8                         (access) /WinDump.exe
 9 Wed Oct 27 2004 16:23:50    (modify) /WinPcap_3_1_beta_3.exe
10 Wed Oct 27 2004 16:23:54    (change) /WinPcap_3_1_beta_3.exe
11                         (change) /WinPcap_3_1_beta_3.exe
12 Wed Oct 27 2004 16:23:56    (modify) /WinPcap_3_1_beta_3.exe
13 Wed Oct 27 2004 16:24:02    (modify) /WinDump.exe
14 Wed Oct 27 2004 16:24:04    (change) /WinDump.exe
15                         (change) /WinDump.exe
16 Wed Oct 27 2004 16:24:06    (modify) /WinDump.exe
```

Now we see Mr Lawrence saving the packet sniffer utility (WinDump) and the dependant library (winpcap) to the image, likely as part of downloading it. If we remember the peculiar double-file entry behaviour seen in the analysis above, it is likely this was saved from within an application of some sort, probably a web browser (as part of the download).

At the very least, at this point we can surmise that there is intent on Mr Lawrence's part to gain access to private or restricted information or communications by downloading and running the sniffer.
In particular, given Mr Lawrence's interest in Ms Conlay and the evidence we have suggesting her rejection of his advances (most importantly, the contents of "hey.doc"), there is enough information to believe that he will be attempting to gain some information about Ms Conlay that may advance his cause.

```
17 Thu Oct 28 2004 00:00:00    (access) /coffee.doc
18                         (access) /_ap.gif
19                         (access) /_apture
20                         (access) /WinDump.exe
21                         (access) /WinPcap_3_1_beta_3.exe
22                         (access) /_ap.gif
```

Lines 20-21 above indicate the access of the packet sniffer executables among other things; this access is likely to be part of the installation process, not shown here as the sniffer utility was likely installed on a local disk on Mr Lawrence's desktop computer, which we do not have access to.

```
23 Thu Oct 28 2004 11:08:24    (change) /_apture
24 Thu Oct 28 2004 11:11:00    (modify) /_apture
```

Lines 23-24 show the packet capture being saved to the device. At this point, Mr Lawrence has run the sniffing utility and actually captured packets. Given the contents of the packets, particularly the contents of the email Leila (Ms Conlay) sent to "SamGuarillo@hotmail.com", we can say that Mr Lawrence has undisputedly violated Ms Conlay's privacy as well as one or more legal statutes.

```
25 Thu Oct 28 2004 11:17:44    (change) /_ap.gif
26                        (change) /_ap.gif
27 Thu Oct 28 2004 11:17:46    (modify) /_ap.gif
28                        (modify) /_ap.gif
```

The lines above suggest that Mr Lawrence has analysed the packet capture he saved above and (at least) recognised the contents of the mail message. Further, he has now looked up the location specified by Ms Conlay in the mail - "the coffee shop on the corner Hollywood and McCadden" - on an online service and saved the map to the storage device.

```
29 Thu Oct 28 2004 19:24:46    (change) /coffee.doc
30 Thu Oct 28 2004 19:24:48    (modify) /coffee.doc
```

This final document, authored after the time of the incident at the coffee shop (as reported by Ms Conlay) reveals Mr Lawrence's intent: to find out where Ms Conlay would be and orchestrate some type of meeting or "running-in" that appeared to be a coincidence. Our theory above - that Mr Lawrence was attempting to further his cause (i.e. gaining favour with Ms Conlay, however misguided his methods were) is strengthened. Given the tone of the final mail and the apparent coincidence - taking into account that Ms Conlay is unlikely to have known Mr Lawrence intercepted her communications - it is unsurprising that Ms Conlay was alarmed by Mr Lawrence's behaviour.

3.2 Analysis summary and linking it to case details/statements

Returning to the case information, which states:

> On the afternoon of Friday October 29th, Leila contacted corporate security, stating she was being harassed by Robert Lawrence. Leila stated that Robert has made numerous attempts to meet her, both during and outside of work. Leila also stated that Robert has contacted her at her personal email address, and that his emails have become increasingly aggressive. On the evening of Thursday October 28th, Leila was at a coffee shop with a friend when Robert appeared.

we can say that the evidence and our analysis supports the allegations in the statement. In particular:

- we can tie the USB drive to Mr Lawrence: multiple documents on it have metadata indicating the author was "Robert Lawrence", and conversely, there are no documents by other authors. This is in addition to the USB drive having been found on Mr Lawrence's desk.
- the documents authored by Mr Lawrence support Ms Conlay's statements about Mr Lawrence's multiple attempts to meet her
- the documents suggest increasing aggression by Mr Lawrence as Ms Conlay rejects the offers including an allusion to further aggression ('I heard rumors about a "bad batch" of coffee, hope you don't get any')
- the sequence of events, supported by the data on the storage device, of:
  - capture (and save) network traffic
  - lookup location of coffee shop indicated in network traffic, and save results of lookup/image

indicate that Mr Lawrence knew where Ms Conlay would be on the evening of 28 October
- the combination of Mr Lawrence's demonstrated interest in Ms Conlay and Mr Lawrence's knowledge of Ms Conlay's meeting with "SamGuarillo@hotmail.com" on the evening of 28 October agrees with Ms Conlay's statement that Mr Lawrence was at the coffee shop that evening. In addition, we can say with some certainty that he was there specifically to "bump into" Ms Conlay.
- the dates mentioned in the above statement are supported by the evidence and analysis of it
- there is no other information uncovered in our analysis that contradicts Ms Conlay's statement

Hence we can say that the evidence does support Ms Conlay's allegation that Mr Lawrence has been harassing her. In addition, we have proof that Mr Lawrence has broken one or more statutes in the course of his actions relating to this case.

4. Legal Implications

We have stated above that we have evidence indicating Mr Lawrence has broken one or more statutes during this incident. Specifically, in Australian federal law, there are two separate areas/issues to consider:

1. unauthorised interception of communications
2. possession and supply of interception software

4.1 Unauthorised interception of communications

Mr Lawrence's actions in capturing traffic, including Ms Conlay's communications, clearly fall under the Australian Telecommunications (Interception) Act 1974 [TELINT]. Part IA, section 6, subsection 1, states:

> For the purposes of this Act, but subject to this section, interception of a communication passing over a telecommunications system consists of listening to or recording, by any means, such a communication in its passage over that telecommunications system without the knowledge of the person making the communication.

Part II, section 7, subsection 1 states:

> A person shall not:
> (a) intercept;
> (b) authorize, suffer or permit another person to intercept; or
> (c) do any act or thing that will enable him or her or another person to intercept;
> a communication passing over a telecommunications system.

Obviously, none of the provisions for lawful interception mentioned in the act apply in this case, so Mr Lawrence would be liable for the punishment listed in part X, Section 105 of the Act: a maximum of two years imprisonment, or six months if the matter is heard in a court of summary jurisdiction (e.g. a local Magistrate's court)

4.2 Possession and supply of interception software

In recent times, the Australian government has been moving to legislate areas related to "cybercrime" in an attempt to fix laws that where generally created before the pervasive use of computing and communications technology. In particular, the Crimes Act 1914 [CRIMES] has been updated to deal with the concept of tools that can be used to intercept communications. Section 85ZKB of the Act states:

> (1) A person shall not:
>
> (a) manufacture;
>
> (b) advertise, display or offer for sale;
>
> (c) sell; or
>
> (d) possess;
>
> an apparatus or device (whether in an assembled or unassembled form) that the person knows is an apparatus or device of a kind that is capable of being used to enable a person to intercept a communication in contravention of subsection 7(1) of the Telecommunications (Interception) Act 1979.
>
> Penalty: Imprisonment for 5 years.

Whilst the concept of possession of the interception tools has been likened to possession of tools required for a burglary, it is unclear exactly how this analogy works with (intangible) software. Even more bizarre is the idea that supply, for example, publishing on a website, mirroring, etc., of such tools is an offence (as can be read into the above law) - given that these kinds of tools have and may be used for many types of network or communications research and diagnosis.

Needless to say, it appears that the statutes above, relating to possession and supply of communications interception devices, have not been tested in courts, so the best we can say is that Mr Lawrence may have had additional penalties levied against him had he committed his offences in Australia.

5 Recommendations

5.1 Case-specific recommendations

At this point in the investigation, we need to consider the main non-technical issues relating to this incident, namely, that it is almost certain that:

1. Mr Lawrence has broken the law
2. Mr Lawrence has likely broken the organisation's IT policies - at worst, hopefully as a consequence of 1
3. Mr Lawrence has caused distress to another employee

Regarding aspect 1, we'd need to keep in mind that Mr Lawrence has not caused any damage to property or physical life and has not committed a serious crime or crime against the government, so the activity is unlikely to be of interest to law enforcement. It would be up to management or legal counsel, along with company policies, whether the matter is referred to law enforcement. Aspect 2 is serious, but again, Mr Lawrence has not caused the company any loss (so far) - as long as Mr Lawrence is given some reprimand that might scare off other potential offenders, we might ignore this aspect.
Hence, it is probably best (from an organisational point of view) to focus on aspect 3: Ms Conlay may see the company as negligent in providing a comfortable workplace environment given Mr Lawrence's harassment in the workplace and his use of company

resources (email) as part of this harassment. There is also (though Ms Conlay does not yet know it) Mr Lawrence's violation of her privacy.

Realistically, Human resources and Legal departments would need to be informed of the issues and presented the evidence we have gathered. In concert with management and company policies, they would need to make a decision on how to proceed.

There is, of course, one remaining obvious weak point in our investigation: we cannot yet tie the USB device found on Mr Lawrence's desk to Mr Lawrence himself. He can, after all, insist that the device was not his and has been “planted”. If the case were to go to the courts, we'd want to eliminate this weakness in our case.

Presuming legal counsel or others asked us to proceed with getting evidence tying Mr Lawrence to the USB device, we could proceed as detailed below, at all times getting permission for our activities and consulting with those in authority.

Initially, we'd take advantage of the fact that the USB device is still in our possession and has not been returned to Mr Lawrence. Hopefully, Mr Lawrence does not know anything other than that the device is missing; he should also not be too worried about the contents because - to his (untrained) eyes - the evidence suggesting he has broken any laws has been lost with the deletion of the sniffing tools and capture file.

So, we would request that the lost property or similar department send out an email indicating that a "lost" USB device has been found by the cleaners and requesting the owner come and pick it up. With luck, Mr Lawrence will turn up and claim the device as his own. If he does not, we can take the device to him, claiming that the cleaner remembered seeing the device near his desk and handing it to him. Ideally, he will now claim it as his own. Even if he does not, we can proceed with the next phase: an interview. We would conduct such an interview with at least one IT and one legal person present.

In the interview, we would initially take great pains to put Mr Lawrence at ease and bring him into our confidence. To do this, we may want to make up a story about an allegation that someone has been stealing company secrets and sending them to a competitor, stressing that we know Mr Lawrence is not involved, and asking him if he would help us with our investigations. Most people would be relieved that they are not being investigated themselves, and would certainly not want to arouse suspicion by refusing such a request, so we can then proceed with asking Mr Lawrence a few questions after stating that we think someone may be using a (possibly stolen) USB device to get the secrets out of the company's premises.
If Mr Lawrence had previously not indicated the USB device was his own, we would suggest that the MS Word documents on it had author information indicating they were his, while again stressing that he was not a suspect in this issue. Again, as Mr Lawrence would not want to arouse suspicions, he would likely admit the device was his own, at which point we could ask a series of questions ascertaining if he knew where the device was on a number of occasions, including the evening of 28 October. With luck, he would confirm that he was in possession of the device on that evening, at which point we would have the strongest evidence possible that he did actually commit the offences (and, evidence that refutes the "plant" argument). We'd round off this interview with some final diversionary questions asking if he'd seen any suspicious activity on the evening of 28 October, or on any other evening he'd been working back late.

At this point, our part in this case is over, and all the evidence and investigation findings

would be turned over to legal counsel or whoever was leading the investigation.

5.2 General recommendations

From an Information Security point of view, there are two lessons that we may want to learn from this incident. The first is related to policy. we should be asking ourselves whether our policies governing use of IT/Communications resources were appropriate for this and similar cases, if they need to be strengthened (and how), whether they've been communicated to employees effectively, and whether we have the power and resources to ensure they are adhered to.

We may also want to tie that in to legal statues - that is, informing employees that there are laws about telecommunications interception, for example.

The second lesson is related to IT/Communications infrastructure: namely, is it appropriate to have network infrastructure that is susceptible to interception by relatively unskilled individuals? Although it is unclear from the evidence and analysis whether the interception occurred over a wireless network, we would want to ensure that all wireless networks were audited and verified as secured with some form of encryption (e.g. WPA or better) to stop further interception incidents. If the interception occurred due to hubs being used, we'd definitely recommend that they be replaced with switches in all cases.
As part of the above audit, we'd also insist that relevant documentation or policies relating to the installation of new network infrastructure also be updated to reflect the use of more secure technologies or techniques.

6 Additional Information

Readers may find the following resources, used as part of research for this paper, useful:

Name: Microsoft developer network site
Location: <http://msdn.microsoft.com/>
Description: online developer documentation and articles for all Microsoft products
Use: handy reference for all things windows, particularly win32 API and filesystem documentation

Name: Microsoft knowledge base
Location: <http://support.microsoft.com/kb/>
Description: online access to issues, problems and resolution for all Microsoft products
Use: contains useful overview of filesystems and OS behaviour under certain conditions (also known as "bugs" or "features")

Name: SleuthKit site
Location: <http://www.sleuthkit.org/sleuthkit/>
Description: official site for the sleuthkit forensics tools
Use: general information and useful documentation about the toolkit
Name: Electronic Frontiers Australia
Location: <http://www.efa.org.au/>
Description: website of EFA, "a non-profit national organisation representing Internet users concerned with on-line freedoms and rights" [http://www.efa.org.au/AboutEFA/]
Use: good overview information and criticism about various bits of legislation related to surveillance, communications and privacy ("Quis custodiet ipsos custodes?")

Name: Australasian Legal Information Institute
Location: <http://www.austlii.edu.au/>
Description: provides free Internet access to Australasian legal materials
Use: searchable databases/indexes of all kinds of Australian law

Name: Commonwealth of Australia Law site
Location: <http://www.comlaw.gov.au/>
Description: legal information retrieval system owned by the Australian Attorney-General's Department
Use: online access to Australian Commonwealth acts in various formats

7. List of references

[**FATSPEC00**] Microsoft Corporation, "Microsoft Extensible Firmware Initiative FAT32 File System Specification", Version 1.03, published: 2000-12-6, <http://www.microsoft.com/whdc/system/platform/firmware/fatgen.mspx>

[**SLKTDOC**] Carrier, Brian et al, "Sleuthkit documents", Sleuthkit website, accessed: 2005-02-15, <http://www.sleuthkit.org/sleuthkit/docs.php>

[**TELINT**] Australian Commonwealth, "Telecommunications Interception Act (1974)", including amendments up to Act No. 148 of 2004, published on: 2005-01-12, <http://www.comlaw.gov.au/comlaw/Legislation/ActCompilation1.nsf/0/0B14824B1125A44CCA256F860015FF1B?OpenDocument>

[**CRIMES**] Australian Commonwealth, "Crimes Act (1914)", including amendments up to Act No. 127 of 2004, Volume 2, published: 2004-12-22, <http://www.comlaw.gov.au/comlaw/Legislation/ActCompilation1.nsf/0/D74B4B29399F61C3CA256F8B0083588E?OpenDocument>

[**SKUALAN05**] Alan, <surname unknown>, "two directory entries for same deleted file in FAT16", Sleuthkit-users mailing list, posted: 2005-02-27, accessed: 2005-03-02, <http://sourceforge.net/mailarchive/message.php?msg_id=11020016>

[**SKUCARRIER05**] Carrier, Brian, "Re: two directory entries for same deleted file in FAT16",Sleuthkit-users mailing list, posted: 2005-02-28, accessed: 2005-03-02, <http://sourceforge.net/mailarchive/message.php?msg_id=11032375>

Appendix A: listing of timeline.mactime.out

```
Mon Oct 25 2004 00:00:00    19968 .a. -/-rwxrwxrwx 0        0        3        /her.doc
Mon Oct 25 2004 08:32:06    19968 ..c -/-rwxrwxrwx 0        0        3        /her.doc
Mon Oct 25 2004 08:32:08    19968 m.. -/-rwxrwxrwx 0        0        3        /her.doc
Tue Oct 26 2004 00:00:00    19968 .a. -/-rwxrwxrwx 0        0        4        /hey.doc
Tue Oct 26 2004 08:48:06    19968 ..c -/-rwxrwxrwx 0        0        4        /hey.doc
Tue Oct 26 2004 08:48:10    19968 m.. -/-rwxrwxrwx 0        0        4        /hey.doc
Wed Oct 27 2004 00:00:00   485810 .a. -/-rwxrwxrwx 0        0        7        /
WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
                           450560 .a. -/-rwxrwxrwx 0        0        12       /
WinDump.exe (_INDUMP.EXE) (deleted)
Wed Oct 27 2004 16:23:50   485810 m.. -/-rwxrwxrwx 0        0        10       /
WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
Wed Oct 27 2004 16:23:54   485810 ..c -/-rwxrwxrwx 0        0        10       /
WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
                           485810 ..c -/-rwxrwxrwx 0        0        7        /
WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
Wed Oct 27 2004 16:23:56   485810 m.. -/-rwxrwxrwx 0        0        7        /
WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
Wed Oct 27 2004 16:24:02   450560 m.. -/-rwxrwxrwx 0        0        14       /
WinDump.exe (_INDUMP.EXE) (deleted)
Wed Oct 27 2004 16:24:04   450560 ..c -/-rwxrwxrwx 0        0        14       /
WinDump.exe (_INDUMP.EXE) (deleted)
                           450560 ..c -/-rwxrwxrwx 0        0        12       /
WinDump.exe (_INDUMP.EXE) (deleted)
Wed Oct 27 2004 16:24:06   450560 m.. -/-rwxrwxrwx 0        0        12       /
WinDump.exe (_INDUMP.EXE) (deleted)
Thu Oct 28 2004 00:00:00    19968 .a. -/-rwxrwxrwx 0        0        18       /coffee.doc
                             8814 .a. -/-rwxrwxrwx 0        0        17       /_ap.gif
(deleted)
                            53056 .a. -/-rwxrwxrwx 0        0        15       /_apture
(deleted)
                           450560 .a. -/-rwxrwxrwx 0        0        14       /
WinDump.exe (_INDUMP.EXE) (deleted)
                           485810 .a. -/-rwxrwxrwx 0        0        10       /
WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
                             8814 .a. -/-rwxrwxrwx 0        0        16       /_ap.gif
(deleted)
Thu Oct 28 2004 11:08:24    53056 ..c -/-rwxrwxrwx 0        0        15       /_apture
(deleted)
Thu Oct 28 2004 11:11:00    53056 m.. -/-rwxrwxrwx 0        0        15       /_apture
(deleted)
Thu Oct 28 2004 11:17:44     8814 ..c -/-rwxrwxrwx 0        0        16       /_ap.gif
(deleted)
                             8814 ..c -/-rwxrwxrwx 0        0        17       /_ap.gif
(deleted)
Thu Oct 28 2004 11:17:46     8814 m.. -/-rwxrwxrwx 0        0        16       /_ap.gif
(deleted)
                             8814 m.. -/-rwxrwxrwx 0        0        17       /_ap.gif
(deleted)
Thu Oct 28 2004 19:24:46    19968 ..c -/-rwxrwxrwx 0        0        18       /coffee.doc
Thu Oct 28 2004 19:24:48    19968 m.. -/-rwxrwxrwx 0        0        18       /coffee.doc
```

Appendix B: Double file entries

As described in the analysis, the curious symptom of “double file entries” was first seen in the **`fls`** output as two directory entries for certain files. As further investigation did not reveal any reason for this, a search of various mailing lists and websites led to the sleuthkit-users mailing list, where there was a posting about a very similar situation[12] the poster[13] had seen. Brian Carrier responded[14] with confirmation that he too had seen the behaviour, and also indicated that it had occurred on Windows XP systems and only when files were saved from within applications, rather than when creating them in any other way (e.g. via Windows Explorer or from the command line using "copy con <filename>").

I then performed my own tests on this, mostly to confirm the above conditions for the behaviour, and found that my tests did agree with the reported behaviour. In addition, I tested this on both a Windows 2000 and Windows 98 installation (with FAT16). The former exhibited the behaviour whilst the latter did not.
We can only guess that the Windows 9x and Windows NT (2000, XP) OSs, having different codebases, also have different FAT16 drivers, with the Windows NT family having a bug that shows up under these circumstances. Alternatively, the Windows 9x family may also have the double-file entry bug, but a different directory allocation scheme that means the second file entry always uses the first (deleted) entry instead of allocating a new one.

12 one could be forgiven for thinking the poster was also writing their GCFA practical
13 See [SKUALAN05]
14 See [SKUCARRIER05]

Appendix C: dirty word list

Robert
Lawrence
CC Terminals
Leila
Conlay
date
coffee
sniffer
packet
capture
dump
flowergirl96
Sam
Guarillo
Hollywood
McCadden
hotmail.com
wireless
access point
october
pcap
her.doc
hey.doc
WinPcap_3_1_beta_3.exe