



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Advanced Incident Response, Threat Hunting, and Digital Forensics (Forensics
at <http://www.giac.org/registration/gcfa>

**GCFA
Certification
Practical Assignment 2.0**

**Michael Halm
March 1, 2005**

© SANS Institute 2000-2005, Author retains full rights.

Table of Contents

<u>Table of Contents</u>	2
<u>Bookkeeping</u>	3
<u>Summary Findings</u>	4
<u>Methodology</u>	5
<u>Fsstat</u>	7
<u>Timeline Creation and Analysis</u>	8
<u>Strings</u>	11
<u>File Listing</u>	11
<u>Recovering Files from the Image</u>	12
<u>her.doc</u>	13
<u>hey.doc</u>	13
<u>coffee.doc</u>	13
<u>WinPcap 3 1 beta 3.exe</u>	14
<u>_apture</u>	17
<u>_ap.gif</u>	18
<u>WinDump.exe</u>	18
<u>Recommendations</u>	20
<u>Investigation Round 2</u>	20
<u>SNMP packets</u>	23
<u>Recommendations, Round 2</u>	24
References.....	29
For Further Information.....	29
Appendix A, Contents of Timeline.all.....	29

Bookkeeping

This investigation was conducted primarily on a Gateway laptop with 500 MB RAM. The operating system was Fedora Core 3. Aside from standard Linux command line utilities such as **file** and **strings**, most of the tools used were from The Sleuthkit, which is available at <http://www.sleuthkit.org/>. Autopsy was installed, but because the image being investigated was so small and simple it was never used.

I have followed some formatting conventions for clarity's sake. Very short command line strings may be in quotation marks –“c:\>**md5sum** file.doc.” These were entered into the text of this document manually. Most command line input and output is in text boxes. In all cases, these were copied directly from the terminal window. During an investigation, it is my habit to use a terminal window with a large buffer and copy everything from this window into a text file. I find this more dependable than counting on my memory or notes alone.

The contents of text documents, when not in an image of a screenshot, are in `courier new font, 10 points`. Command line tools I used are in bold type - **strings**, **md5sum**. Special attention should be given to the names of files extracted or recovered from the USB image. Some of them have common generic names such as “map” and “capture.” Sometimes I must compare these files to other files of the same name. Throughout this document when I am referring to a file extracted or recovered from the USB image, the filename will be underscored. If the word is not underscored, then it is either a file not obtained from the USB drive, or it is not being used as a filename at all.

At times it was necessary to test code on a Windows box. For this purpose a generic PC was installed with a fresh copy of Windows XP, patched up to date, with no extra software installed aside from Antivirus. This same box was used to perform test captures of network traffic.

Summary Findings

Leila Conlay has complained of receiving a series of increasingly disturbing e-mails from Robert Lawrence. In addition he turned up in a coffee shop under circumstances which warranted an investigation by corporate security. Corporate security gave me a USB drive to examine in support of their investigation. So what does the content of the USB drive have to do with the complaint?

First there are three Word documents on the drive which seem to contain the bodies of the e-mails Leila received. The first, written on the morning of Monday, Oct 25th, is friendly in tone. The second, written the next morning, is defensive. The third document, written on Thursday evening, is creepy and threatening. Hidden data that MS Word embeds in documents shows that these were written using a copy of MS Word registered to Robert Lawrence.

Examination of the USB drive also found four deleted files, three of which were recoverable. Two of these files were programs which together enable the user of a Windows computer to perform a wiretap of network communications. These programs were used to intercept communications that Leila Conway sent on Thursday Oct 28, including an e-mail, and store it in a capture file. This capture file was also recovered from the USB drive. In the intercepted e-mail, Leila arranges to meet a friend at a coffee shop. The final deleted file on the USB drive is a map showing the location of this coffee shop.

The evidence suggests but does not prove that Robert Lawrence is responsible for all of these files, and that he is guilty of performing an illegal wiretap, and that he used the information thus gathered to intrude on Leila Conlay's date at the coffee shop.

However, there are a number of problems and inconsistencies with this theory. None are fatal to it, but further investigation is necessary.

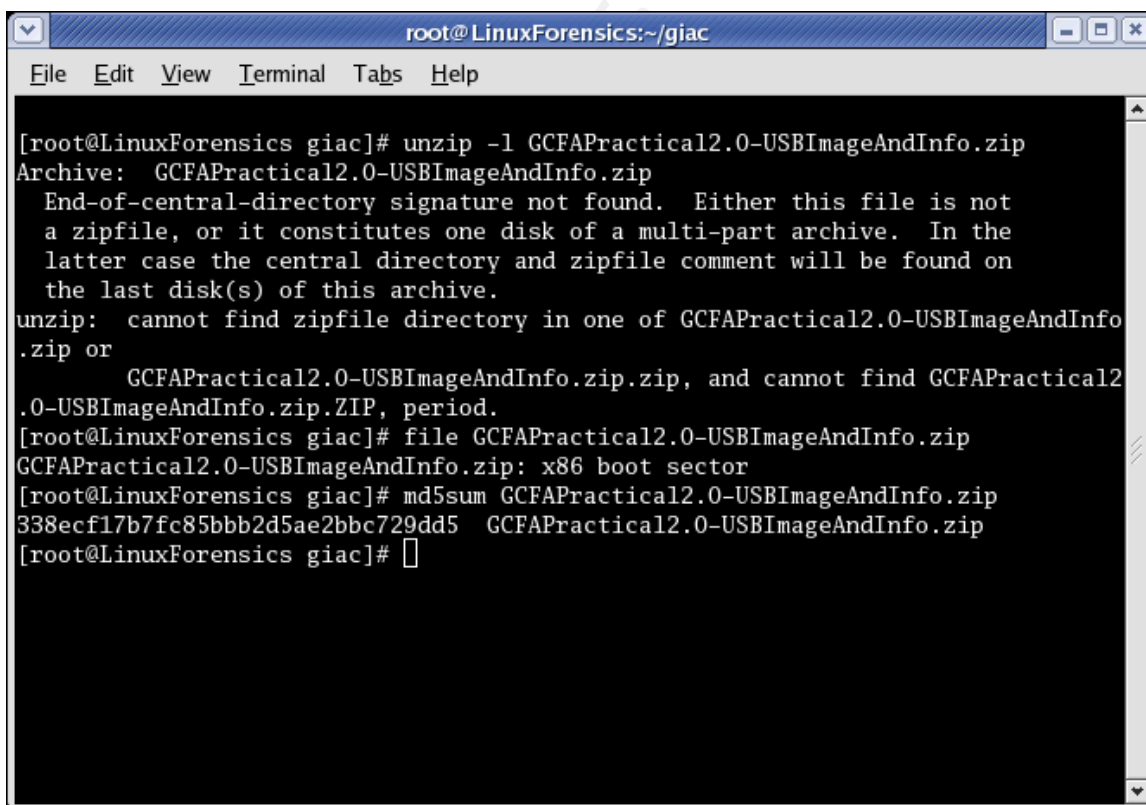
Finally, there is evidence suggestive, but not conclusive that an attempt was made to tamper with the router in the sales office.

Methodology

I downloaded the gzipped file GCFAPractical2.0-USBImageAndInfo.zip.gz from the SANS web site. I ran `gunzip -l` on the archive to find out what files it contained and also as a preliminary check that the download was successful. The archive contained a single zipped file – GCFAPractical2.0-USBImageAndInfo.zip. I extracted the zip file. I then ran `unzip` on the file, but got an error message.

I ran the **file** program on GCFAPractical2.0-USBImageAndInfo.zip. **File** is a program that reads a file's header and footer and compares them to a rather extensive database of known file forms. It is a good first step when attempting to identify an unknown file, or finding out why a “.zip” file won't unzip. **File** gave the following output: “GCFAPractical2.0-USBImageAndInfo.zip: x86 boot sector. “

This suggested that the file was the disk image. I ran **md5sum** on the file. The resultant hash was identical to the hash given in at the end of the case synopsis, as seen in this screenshot:



```
root@LinuxForensics:~/giac
File Edit View Terminal Tabs Help

[root@LinuxForensics giac]# unzip -l GCFAPractical2.0-USBImageAndInfo.zip
Archive:  GCFAPractical2.0-USBImageAndInfo.zip
  End-of-central-directory signature not found.  Either this file is not
  a zipfile, or it constitutes one disk of a multi-part archive.  In the
  latter case the central directory and zipfile comment will be found on
  the last disk(s) of this archive.
unzip:  cannot find zipfile directory in one of GCFAPractical2.0-USBImageAndInfo
.zip or
       GCFAPractical2.0-USBImageAndInfo.zip.zip, and cannot find GCFAPractical2
.0-USBImageAndInfo.zip.ZIP, period.
[root@LinuxForensics giac]# file GCFAPractical2.0-USBImageAndInfo.zip
GCFAPractical2.0-USBImageAndInfo.zip: x86 boot sector
[root@LinuxForensics giac]# md5sum GCFAPractical2.0-USBImageAndInfo.zip
338ecf17b7fc85bbb2d5ae2bbc729dd5  GCFAPractical2.0-USBImageAndInfo.zip
[root@LinuxForensics giac]#
```

This is a good time for a quick explanation of md5sums - what they are and why we use them. The md5sum algorithm is described in detail in

RFC1321.¹ It was created to confirm the identity of files for digital commerce. The algorithm divides the file into portions, performs complex one-way mathematical functions on the portions of the file and recombines the results to create a digital “signature” for the file. The likelihood of two files having identical md5sum signatures is sufficiently remote (2^{64}) that US courts have accepted matching md5sum signatures as proof that two files are identical. For this reason, the first procedure with any file that may be evidence is to take an **md5sum** of the file. The last procedure in examining such a file is to take another **md5sum** and compare it to the first. Altering a single bit of the file will result in the sums not matching. If the sums match, we know we have not inadvertently altered the evidence.

GCFAPractical2.0-USBIImageAndInfo.zip is the disk image. It is the entire contents of the physical hard drive (including areas that contain no files) copied bit by bit and stored as a file. However, computers do not store files directly on a raw disk. First they create a section or sections (partitions) on the raw disk and prepare these partitions (format them) to store files. Our next step is to identify and extract any data partitions in this raw image for examination. **Mmls** is a program that will examine a physical disk or raw disk image and identify the types and locations of partitions. **Mmls** must be told to search for partitions of a given type (operating system). If you tell it to search for partitions of an operating system for which no partitions are present it will simply return an error message. Given the prevalence of Windows, especially on business workstations, I will try DOS first.

```
[root@LinuxForensics giac]# mmls -t dos GCFAPractical2.0-USBIImageAndInfo.zip
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)

Here at the most basic level of organization, the disk is simply measured into sectors – physical portions of the disk able to hold, in this case, 512 bytes of data. The image has a DOS (FAT16) partition on sectors 32-121950. We can ignore the Primary Table and unallocated space unless we find evidence that someone with very sophisticated technical knowledge and tools had something to hide there. Any files stored on this disk should be within the DOS FAT16 partition.

DD is a tool that will copy all or part of a physical disk or disk image bit by bit, sector by sector. Copy or backup commands generally operate at the file level – they copy file by file. It is important for investigative purposes to copy every single bit in a partition including, and perhaps especially, those that are not part of a file. We have to tell **dd** what sectors to copy and what size those sectors are. We also have to tell it where to copy from and where to store the copied sectors. The **bs=512** switch tells **dd** that sectors are 512 bytes in length. **Skip=32** tells it not to copy the first 32 sectors (sectors 0 through 31). **Count** tells it how many sectors to copy. We gather all this information directly from the output of **mmls**. Input file (if) will be **GCFAPractical2.0-USImageAndInfo.zip** and we will call the destination file (of) **usbpart.img**. Immediately after making the image, we will run an **md5sum** on it, so we can verify that none of our subsequent procedures alters the evidence.

```
[root@LinuxForensics giac]# dd if=GCFAPractical2.0-USImageAndInfo.zip bs=512 skip=32
count=121919 of=usbpart.img
121919+0 records in
121919+0 records out
[root@LinuxForensics giac]# md5sum usbpart.img
5f830a763e2144483f78113a8844ad52  usbpart.img
```

This resulti

T
h
i
s
r
e
s
u
l
t

ulting image file is an image of a dos16 formatted partition. Formatting a partition creates a blank table of contents or index so that an operating system can store and retrieve files. For a DOS partition, this index is called the File Allocation Table or FAT. Each entry in the table contains certain information. The first byte of data for an unused table entry is a digital marker indicating that this table entry is available to be used. A used table entry will start with the name of the file it is indexing. The first letter of the file's name will overwrite the "available" marker for that table entry. The table entry will also indicate the length of the file, the physical location (block numbers) of the file on the disk and the date and time the file was created, last read and last modified.

We have tools which allow us to mount this file just as we would a partition on a physical disk. However, then the operating system normally would alter the file access times every time we access one of the files in the partition. For our purposes, we want at the least to record these access times before we risk changing them. Instead of mounting the file as a file system, we can use tools which have the capability internally to parse the data structures on a fat16 partition. The Sleuthkit supplies us with a number of tools that are specifically designed to examine such images and understand their file structure without the need to mount the partition and risk altering any data. In fact, **mmls** is from the Sleuthkit.

Fsstat is also from the Sleuthkit. It will give us some basic info about the

file system which may be important. Here is a portion of the output from **fsstat**:

```
METADATA INFORMATION
-----
Range: 2 - 1942530
Root Directory: 2

CONTENT INFORMATION
-----
Sector Size: 512
Cluster Size: 1024
Total Cluster Range: 2 - 60705

FAT CONTENTS (in sectors)
-----
511-550 (40) -> EOF
551-590 (40) -> EOF
591-630 (40) -> EOF
```

We can see that the metadata (FAT) entries number from 2 - 1942530 and that the first entry in the file table, number 2, is the root directory. We also see that content is currently stored in three blocks that fill consecutively sectors 511 through 630.

Timeline Creation

Next we will create a timeline of file activity on this partition. This will require four steps. FAT records the dates each file was created, as well as the last times it was modified or accessed (Modified Accessed Created = MAC times). We will use **fls** to copy these timestamps from the files in the image. Likewise, **ils** can parse all the meta data (file table) and gather the timestamps from *deleted* files. (When a file is deleted from a DOS partition, normally all that is actually lost is the first letter of the filename. It is overwritten with the “available” marker. The rest of the filename, the time stamps, the file length and the physical location of the beginning of the file data remain intact until that particular table entry is over-written to store a new file.) With both of these tools, we use the -f switch to tell the tool what type of partition to parse and the -m switch to tell them to save the time stamps in a format that is useful for creating a timeline. We then use the **cat** command to concatenate the two output files from **fls** and **ils** into a single file. **Mactime** takes the info from this file, and sorts all the timestamps in ascending order, so we get a chronological record of all

```
[root@LinuxForensics giac]# fls -f fat16 -m / -r usbpart.img>usbpart.flx
[root@LinuxForensics giac]# ils -f fat16 -m usbpart.img>usbpart.ils
[root@LinuxForensics giac]# cat usbpart.?ls>usbpart.mac
[root@LinuxForensics giac]# mactime -b usbpart.mac>timeline.all
```

– creation, access and modifications – on the partition.

The full contents of the timeline.all file is included as an appendix. Here is a portion of it.

```
Tue Oct 26 2004 08:48:10      19968 m.. -/-rwxrwxrwx 0      0      4
/hey.doc
Wed Oct 27 2004 00:00:00          0 .a. -rwxrwxrwx 0      0     12
<usbpart.img-_INDUMP.EXE-dead-12>
                                450560 .a. -/-rwxrwxrwx 0      0
12      /WinDump.exe (_INDUMP.EXE) (deleted)
                                485810 .a. -/-rwxrwxrwx 0      0      7
/WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
```

The timeline is very small and consists of about three pages of output just like this sample. It is not as difficult to decipher as it may seem. You simply have to know how to reduce and possibly reformat the output to boil it down to the useful information. The first portion of each entry is a date and time. This is followed by a column that contains the file size, and a column for the type of entry – modified, accessed or created.

The last two columns are the file table entry number and the filename. The remaining middle columns contain access rights, user and group names. The columns are in the output because **mactime** was originally written for Unix file systems. None of these attributes are actually recorded in a FAT system. In order to keep the format of its output consistent, **mactime** reports default values in these columns. They do not represent information present in the FAT and can be ignored.

The second two entries in the above sample have an access time of midnight (00:00). This is another anomaly resulting from the fact that **mactime** was written for Unix and used here on a FAT partition. We will consider these also as spurious entries since our knowledge of this case so far does not give us any reason to believe that relevant file system activity took place at midnight.

There are two kinds of duplicate entries in the timeline. Mactime reports twice for deleted files. You see this above with two entries for file table entry 12, one named `<usbpart.img-_INDUMP.EXE-dead-12>` and the other `/WinDump.exe (_INDUMP.EXE) (deleted)`. From the file table entry number, you can see that

these are the same file. **Mactime** does this with deleted files.

Sometimes we have two separate file table entries for the same file. For instance, Winpcap appears at number 7 in the file table and also at number 10. This double entry will also show up in other file commands that parse the FAT. It is a result of the way Windows XP saves files. For command line copies or drag and drop operations, Windows XP will create a single entry in the FAT. But when a file is saved from the “File, save” menu in an application, Windows XP makes two entries in the FAT. One is a zero length file with no associated data blocks, and the other entry is a normal file table entry². On our partition, file table entries at 7, 12 and 16 are zero length duplicate entries.

Finally, it is useful to know that when a file is first created in a file table, the attributes for created and modified are both set. So the first time a file appears in a timeline, there are two entries within a few seconds of each other for created time and modified time.

If we remove from our timeline.all file only the output in the categories listed above, we get this reduced timeline which is very easy to interpret.

Reduced Timeline

```

Mon Oct 25 2004 08:32:06
19968 ..c          3      /her.doc

Tue Oct 26 2004 08:48:06
19968 ..c          4      /hey.doc

Wed Oct 27 2004 16:23:54
485810 ..c         10     /WinPcap_3_1_beta_3.exe (_INPCA~1.EXE)
(deleted)

Wed Oct 27 2004 16:24:04
450560 ..c         14     /WinDump.exe (_INDUMP.EXE) (deleted)

Thu Oct 28 2004 11:08:24
53056 ..c          15     /_apture (deleted)

Thu Oct 28 2004 11:11:00
53056 m..          15     /_apture (deleted)

Thu Oct 28 2004 11:17:44
8814 ..c           17     /_ap.gif (deleted)

Thu Oct 28 2004 19:24:46
19968 ..c          18     /coffee.doc

```

We see that seven files have been created on this partition in four days. One of the files was modified a few minutes after creation and four of the files were deleted. Now we will examine those files.

Strings

Before we extract individual files from the image, we will run **strings** against it. **Strings** will output all instances on the image where four or more printable characters occur consecutively. Most files, even if they are not text files or documents, contain some bits of text. These bits can help us identify the nature of an unknown file. By running **strings** against the entire image, we will gather evidence from unallocated space as well as currently allocated files. Even if a file from this image is deleted and unrecoverable, the **strings** output can give us evidence from any portions of it that have not been overwritten. If the **fls** command had shown many files and/or a lot of file writing and deletions, we would simplify our results by running **strings** separately on the unallocated space and then on individual files we need to examine in more depth. On this small partition, a single file with all the text on the partition should be manageable. I will use the **-radix** option to have **strings** output the location of each string to aid in associating text strings with their corresponding files. This location will be in the form of a six digit number in the left column. The **strings** output is saved to a file for use later.

File Analysis

Fls will produce a list of file entries in a directory. If the **-r** switch is used, **fls** will run through all child directories of the starting directory. If we don't designate a starting directory, it will start from the root. So if we use **-r** and specify no directory, we get a list of all files in all directories. It will also list deleted files as long as their file table entry has not been reused.

```
[root@LinuxForensics giac]# fls -rp -f fat16 usbpart.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
```

We see that there are three Word documents and several deleted files on the drive. (The files with an asterisk * are deleted files) You will notice that the first character in the name of some of the deleted files is an underscore.

Remember that deleting a file in a FAT partition overwrites the first letter of the filename. In some cases, **fls** has provided us with a reasonable guess at the first letter of a deleted file's name.

The first number in each line of this output is the number of the file table entry. Remember that **fsstat** showed us earlier that the file table started with 2, and that 2 is the root directory entry. The table entries are created in chronological order. In a very simple file system such as this, where there are no subdirectories and very little deletion and overwriting, the numbers of the file table entries give us a pretty good chronology of file activity. This chronology agrees with the timeline we created earlier. Several files were made, some of them deleted, and then one more created. Chances are excellent that all or all but one of the deleted files have not yet been overwritten and can be recovered.

One more item of interest in the output is the fact that three files have two entries each. The files are WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) at file table 7 and 10, WinDump.exe (_INDUMP.EXE) at file table 12 and 14, and _ap.gif at file table 16 and 17. We know that Windows XP creates such entries when a file is created from within an application using the "File, save" command². These three files were saved from within an application on a Windows XP system.

Extracting and recovering files from the image is done with **icat**. **icat** takes a storage device and inode number (on a FAT partition, it takes the file table entry number) as input and returns the file associated with that entry if it is recoverable. The file table entry numbers we get from the output of the **fls** command. Here are the commands used to extract the files from usbpart.img, including one typo.

```
[root@LinuxForensics giac]# icat -r -f fat16 usbpart.img 15 >capture
[root@LinuxForensics giac]# icat -f fat16 usbpart.img 3 >her.doc
[root@LinuxForensics giac]# icat -f fat16 usbpart.img 4 >hey.doc
[root@LinuxForensics giac]# icat -f fat16 usbpart.img 18 >coffee.doc
[root@LinuxForensics giac]# icat -r -r fat16 usbpart.img 10 >winpcap.exe
icat: extXfs_open: open fat16: No such file or directory
[root@LinuxForensics giac]# icat -r -f fat16 usbpart.img 10 >winpcap.exe
[root@LinuxForensics giac]# icat -r -f fat16 usbpart.img 14 >windump.exe
[root@LinuxForensics giac]# icat -r -f fat16 usbpart.img 17 >map.gif
```

You will notice that the command for extracting an existing file (**icat -r**) is identical to the command for recovering a deleted file. Remember, unless the file data was overwritten, all that is lacking in the file table entry is the first letter of the filename. We have to supply the names of the files we are saving. I take

the liberty of naming _apture and _ap.gif capture and map.gif, respectively.

I also took md5sums of all the files. The hash for windump.exe will be useful when we attempt to confirm the identity of that executable.

```
[root@LinuxForensics giac]# md5sum windump.exe
79375b77975aa53a1b0507496107bff7 windump.exe
```

All three Word documents contain short messages. All the messages are consistent with e-mails that Leila Conlay claims she received from Robert Lawrence. Why do we find e-mails as Word documents? Word is the default e-mail editor for Microsoft Outlook as well as several other e-mail programs. However, e-mail programs do not normally save Word copies of the e-mails to disk. One possibility is that these documents were created in Word first and the messages copied into e-mail. There are other theories possible that would meet the facts. For now, we note that this is not expected behavior.

her.doc

The first document, her.doc, contains the message:

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.doc

The second document, hey.doc contained:

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?

coffee.doc

And 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...

In every document MS Word creates, it embeds the identity of the registered owner (the person identified when the copy of Word was first installed). The information can be changed, but it is not visible when the document is opened in Word and therefore most users are unaware of its existence. However, the **strings** utility or a hex editor will reveal it.

All three of these documents list Robert Lawrence as the registered user. This does not by itself prove that Robert Lawrence authored these documents. But a quick check of the computers at CC Terminals will reveal which computer lists Robert Lawrence as the registered user of Word. With this knowledge and the timestamps of the files, we will know exactly when and where these files were created. A screenshot of the owner data from her.doc, as seen in **khexedit**, is shown below.

© SANS Institute 2000 - 2005, Author retains full rights.

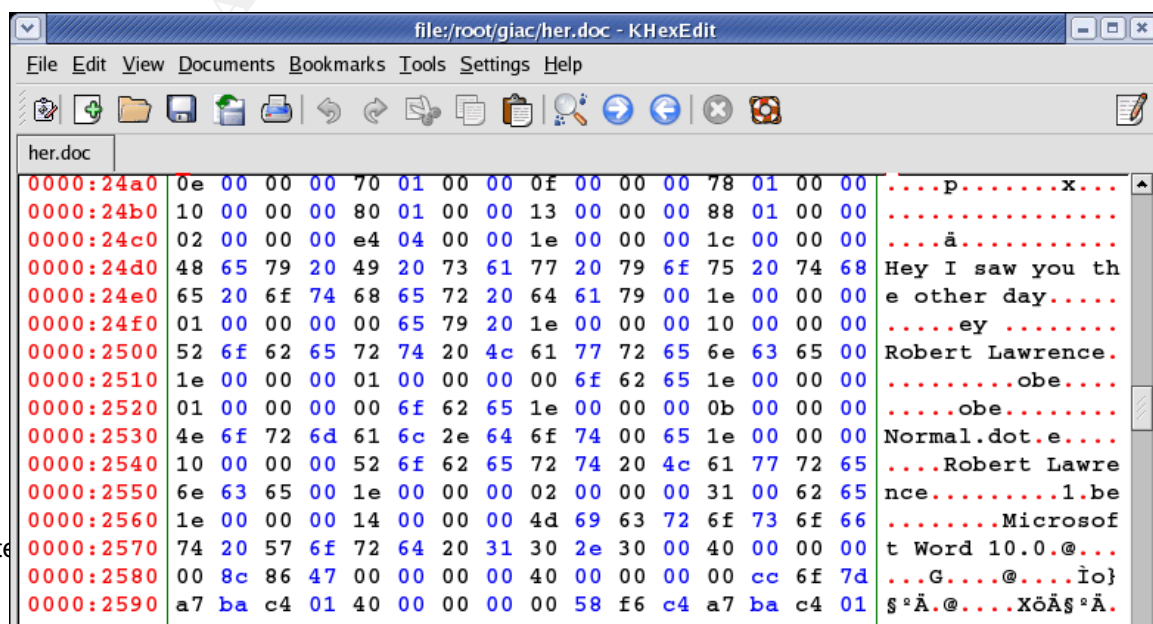
WinPcap 3 1 beta 3.exe

The file WinPcap 3 1 beta 3.exe could not be recovered. **lcat**, in fact, returned a 0 byte file. We can use the **ils** tool to list the file table entries to find out why. Here is a partial result of the output using **ils** to list the file table information for deleted files.

```
root@LinuxForensics giac]# ils -f fat16 usbpart.img
class|host|device|start_time
ils|LinuxForensics|usbpart.img|1107382205
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_mode|st_nlink|st_size|st_block0|st_block1
7|f|0|0|1098919436|1098860400|1098919434|100777|0|0|0|0
10|f|0|0|1098919430|1098946800|1098919434|100777|0|485810|42|0
12|f|0|0|1098919446|1098860400|1098919444|100777|0|0|0|0
```

Earlier, **fls** showed us that WinPcap 3 1 beta 3.exe was stored both at file table entry 7 and file table entry 10. We see here that the file length for table entry 7 (which I have marked in bold type) is zero. The second entry, 10, has a documented length of 485,810 bytes and is stored on disk beginning at block 42 (also marked in bold type). We would normally expect to find the body of this file starting at this location. But the file was deleted. On Thursday evening, Oct 28, coffee.doc was created. Here is the **ils** output for that file:

```
18|a|0|0|1099016688|1098946800|1099016686|100777|1|19968|42|0
```

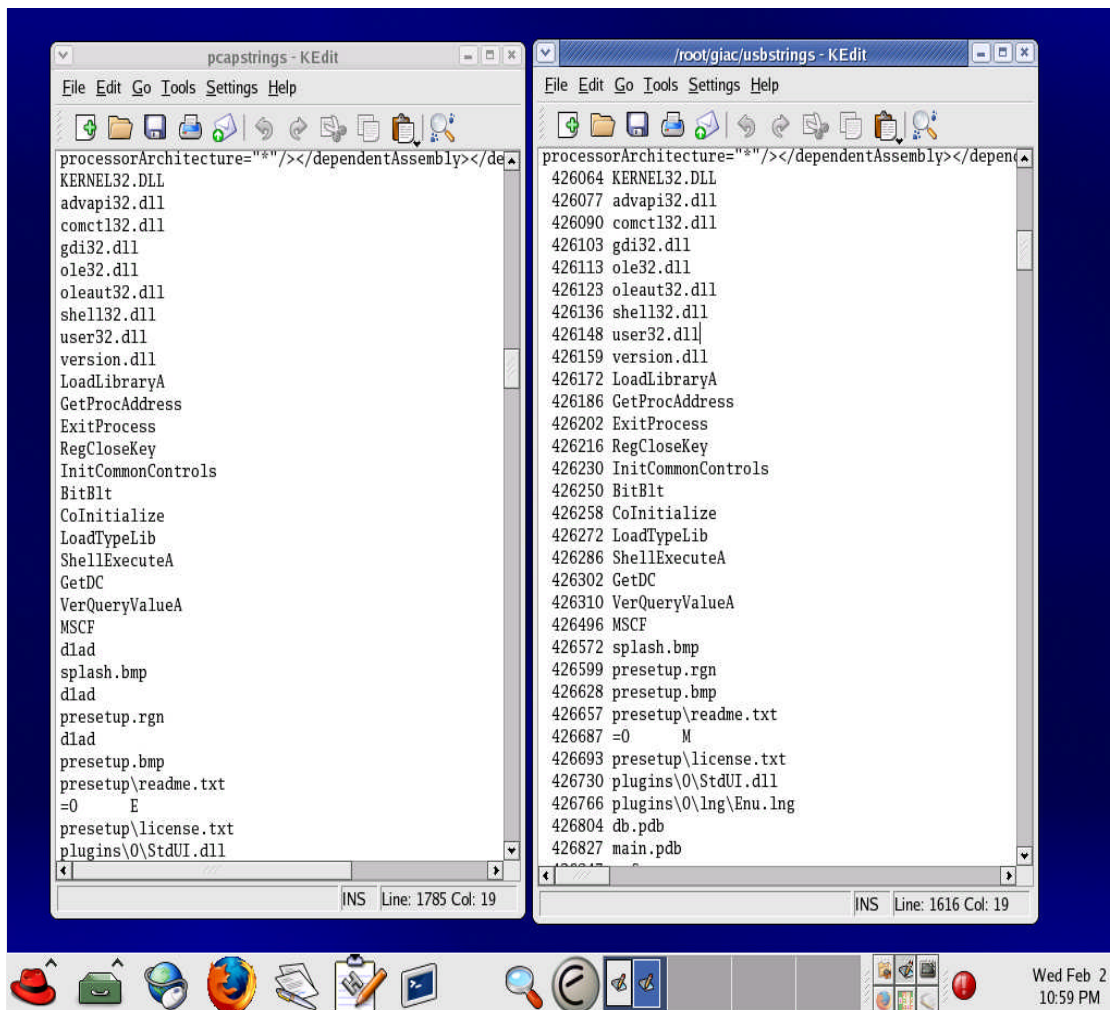


The beginning block for file table entry 18 (coffee.doc) is block 42. So the data for coffee.doc has overwritten at least the beginning part of the WinPcap 3 1 beta 3.exe file. However, coffee.doc was only 19,968 bytes in length, so most of the WinPcap 3 1 beta 3.exe file was not overwritten. If we have an idea what the WinPcap 3 1 beta 3.exe file was, and we are able to obtain another copy of it, we can run **strings** against the copy. We can compare this result to the file we saved earlier containing the strings from the entire USB image file. If we find significant matches, it is almost certainly the same file.

A quick Google search for “winpcap” gives us www.winpcap.polito.it, a page on the website of the Politecnico di Torino. The winpcap home page says “WinPcap is an open source library for packet capture and network analysis for the Win32 platforms. It includes a kernel-level packet filter, a low-level dynamic link library (packet.dll), and a high-level and system-independent library...”³ At this point in time, the currently available version of WinPcap is 3.1 beta 4. Beta 3 has not yet been posted in the archive section of the site and is thus currently unavailable.

However, we can still download the code for beta 4, run strings against it, and compare the output against the strings output from our USB image. We would not expect to find exact matches, given that these are two different versions of the program. But if we find a significant portion of strings output that mostly agrees, we will have very strong evidence that WinPcap 3 1 beta 3.exe was, in fact the same program. Here is a screen shot comparing a portion of the strings output from the downloaded winpcap beta 4 installer to strings found on the USB image. There is, in fact a much longer section of almost identical output. Remember, the extra column of six digit numbers on the right panel are radix “addresses” of these strings in the image – not part of the data itself.

© SANS Institute



WinPcap 3.1 beta 3.exe does seem to be the same program as the winpcap installer from the winpcap.polito.it site. Can we tell if the file was actually run? The executable file is actually an installer for the driver and dll's in the package. If we find these dll's, or any of the registry entries that the installer makes on a computer we will know that the installer was run. The files are wpcap.dll, wanpacket.dll and packet.dll and are installed in the \Windows\System32 directory.

Winpcap does not offer any features directly accessible to a user. The dll's offer a programming interface through which other programs can call upon the functions of the packet driver. So in a practical sense, we don't have to even find out if the installer was run. By itself, it offers no functions to the user.

apture

Next we will examine the apture file. Running **file** against capture (we will refer to this file as capture going forward) we get:

```
[root@LinuxForensics giac]# file capture
capture: tcpdump capture file (little-endian) - version 2.4 (Ethernet, capture length 4096)
[root@LinuxForensics giac]#
```

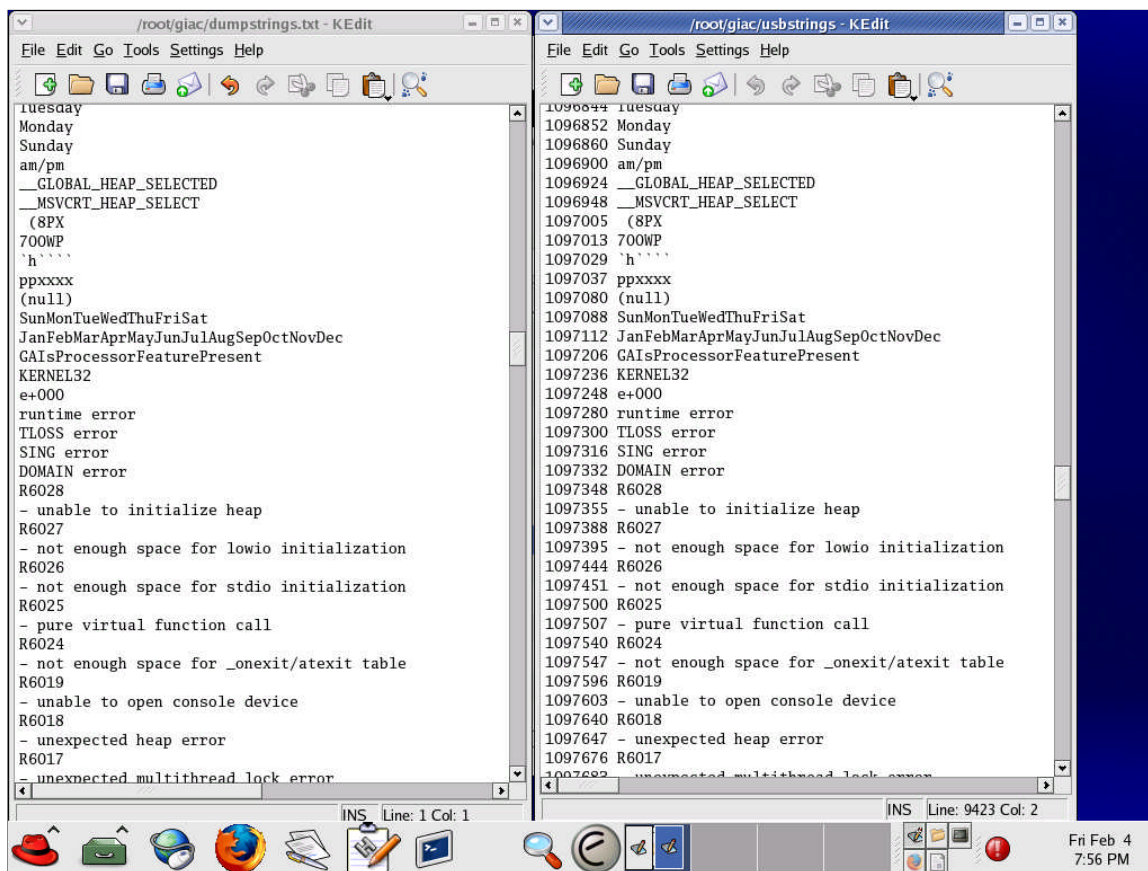
File identifies this as a capture file created by a program called tcpdump. Tcpdump is a unix utility for capturing network traffic. Why does **file** identify capture as a tcpdump capture file? We were expecting this file to be identified as a Windump capture file.

Windump is a win32 port of tcpdump by researchers at the Politecnico di Torino in Italy. The user's manual for Windump is, in fact, the tcpdump man pages with an addendum for the Windows-specific features of windump. Perhaps capture files created by Windump have a header similar enough that the **file** utility will identify them as tcpdump capture files. We will test this hypothesis when we examine the Windump.exe file.

If capture is a network capture file created either by tcpdump or Windump, we will be able to open it with Ethereal. Ethereal does open the file and it does indeed contain network traffic. The first fact that is obvious in looking at the traffic in the capture file is that it is very targeted. Only traffic to or from 192.168.2.104 is represented. If this should turn out to be the network address of the PC on which Windump was installed, then this capture file represents no illegal activity (though the act of installing a sniffer -or any unauthorized program – may violate company policy). But if the address should turn out to belong to anyone else, then an illegal wiretap has been performed.

We will highlight the first packet in the capture and right click and select "Follow TCP Stream." It is common for computer, when communicating over a network, to be carrying on multiple digital sessions (or "conversations") simultaneously. Each conversation is broken up into small parts. The parts (packets) are numbered and sent separately, then reassembled in order at the other end. In a capture such as this, the packets from multiple conversations are mixed together and making sense of them is difficult. When we tell Ethereal to follow a tcp stream, we are instructing it to take all the packets representing a single session and reassemble them. The first stream represented in this capture represents a connection to a Hotmail e-mail server. Here is a portion of that stream.

```
new&to=SamGuarillo@hotmail.com&cc=&bcc=&subject=RE%3A+coffee&body=Sure%2C+coffee+sounds+great.++L
LeilaHTTP/1.1 100 Continue
```



The intended recipient of this e-mail is the owner of the SamGuarillo@hotmail.com account. The note is signed by "Leila." We know that Leila Conway arranged to meet a friend for coffee Thursday evening. Robert Lawrence's unlikely appearance at that same "out of the way" coffee shop at the same time precipitated Leila's complaint to corporate security. This e-mail appears to be when the meeting was arranged.

ap.gif

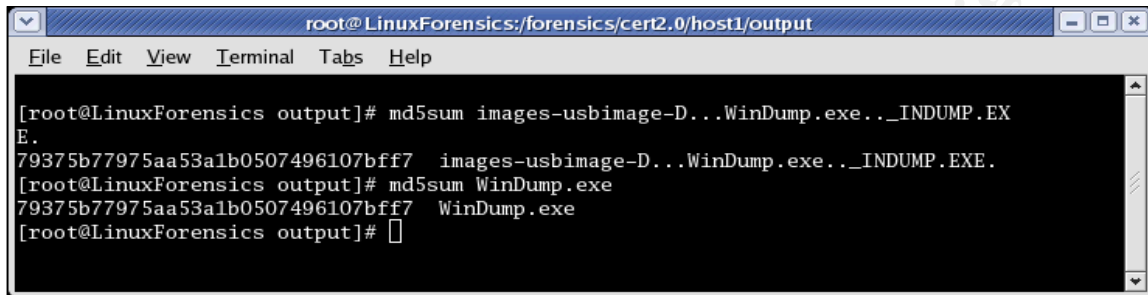
File reveals map.gif to be a gif image file. This is no surprise. Opening the file in a viewer reveals it to be a map showing the location of the coffee shop at Hollywood and McCadden. The file was present on Lawrence's usb drive *after* Leila's e-mail, but *before* the 7 PM meeting at the coffee shop. This suggests that Mr Lawrence did in fact read the e-mail and did not show up at the coffee shop by chance.

WinDump.exe

The WinDump.exe file recovered from the USB drive was 450,560 bytes in length. WinDump.exe can be downloaded from the Politecnico di Torino web site. A copy of Windump.exe downloaded from this site is also 450,560 bytes in

length. Running strings against the recovered WinDump.exe and the downloaded WinDump.exe produces identical results, a portion of which are viewable in this screenshot.

Comparing the checksums also provides identical results.



```

root@LinuxForensics:/forensics/cert2.0/host1/output
File Edit View Terminal Tabs Help

[root@LinuxForensics output]# md5sum images-usbimage-D...WinDump.exe..._INDUMP.EXE
79375b77975aa53a1b0507496107bff7 images-usbimage-D...WinDump.exe..._INDUMP.EXE.
[root@LinuxForensics output]# md5sum WinDump.exe
79375b77975aa53a1b0507496107bff7 WinDump.exe
[root@LinuxForensics output]#

```

So the checksums are identical. In addition, we have identical lengths, names and strings files for these two files. They are the same program. But we will perform a few more tests. We installed WinPcap 3 beta 4 to an XP computer and copied the inDump.exe file recovered from the USB drive to that computer, renaming the file WinDump.exe. Opening a cmd prompt in that directory we were able to run WinDump with the results expected from the manual at <http://windump.polito.it/docs/manual.htm>. "WinDump.exe -D" returned a list of the network adapters on the computer. We then ran "WinDump.exe -i 2 -w testcap" to produce a capture file. This created a capture file called testcap. Running **file** against testcap told us that this was a tcpdump capture file. This is what also what **file** told us about the apture file that was on the USB drive. So we have confirmed our hypothesis that a capture file from Windump will have a header identifying it as a tcpdump capture file.

There is no reasonable doubt that the inDump.exe file on the usb drive is WinDump.exe developed by the Politecnico, that WinPcap was installed on a Windows computer and that electronic communications from Leila Conway's computer were intercepted and recorded to the apture file using the WinDump .exe program. These were stored on a USB drive alongside files that were created using Robert Lawrence's copy of MS Word. That USB drive was found in Robert Lawrence's cubicle the next day.

Recommendations

At this point in the investigation process, I prefer to review a bit. Are there any anomalies in the evidence? Do I have any nagging questions? How clear is the evidence and my reading of it? What would a creative defense attorney do with this evidence?

In this case, there are a few questions in my mind. First, whoever downloaded and ran Windump, created the capture file, and read the e-mail in the captured packets had more technical skill and knowledge than I would expect from a sales associate. So the question bothers me – why didn't this person run a wipe utility on the USB drive. Someone with that level of knowledge would surely know how easy it is to recover deleted files on a FAT partition. Perhaps when someone is committing a crime, he/she could get anxious and make simple mistakes they would not normally make. This is may be the explanation.

However, the mere existence of these files is a problematical. Several e-mail applications can be configured to use Word as the e-mail editor. However none of them will by default save a Word copy of the e-mail at all, much less save it to a USB drive. They will save the e-mail itself in the mailbox. It is possible to open Word and compose a message and then copy the message into the e-mail and then save the now useless Word document to the hard drive and then copy the file to the USB drive. But this is quite a bit of extra work for such short messages as these. It is only worth the effort if Word is not your default editor but you intend to use special formatting or features, such as tables, that are not available in your e-mail editor. No such features are present in these messages. Again, this question is not fatal to the theory that these represent e-mails that Robert Lawrence sent. But it is unnecessarily complicated, and that bothers me.

Investigation Round 2

The next question goes back to the examination of the capture file. After opening the capture file in Ethereal, at one point I sorted the packets by IP address, in order to see how many external hosts were represented. Right at the top were several SNMP packets. Why would a workstation be receiving SNMP packets? Well the packets were actually broadcast – sent to all workstations on the subnet. This explains why these packets were received by a sales representative's workstation. Here is the screenshot of these packets.

No.	Time	Source	Destination	Protocol	Info
2	0.002154	192.168.2.1	192.168.2.255	SNMP	TRAP-V1 1.3.6.1.4.1.3
29	0.282839	192.168.2.1	192.168.2.255	SNMP	TRAP-V1 1.3.6.1.4.1.3
45	0.360401	192.168.2.1	192.168.2.255	SNMP	TRAP-V1 1.3.6.1.4.1.3
53	0.405668	192.168.2.1	192.168.2.255	SNMP	TRAP-V1 1.3.6.1.4.1.3
83	0.555954	192.168.2.1	192.168.2.255	SNMP	TRAP-V1 1.3.6.1.4.1.3
112	0.874097	192.168.2.1	192.168.2.255	SNMP	TRAP-V1 1.3.6.1.4.1.3
1	0.000000	192.168.2.104	64.4.34.250	TCP	2038 > http [SYN] Seq:

File: capture 51 KB P: 113 D: 113 M: 0

How did these packets end up here in this capture file? Some discussion is necessary at this point on the mechanics behind networks and captures. The vast majority of business workstations are connected to an ethernet network by means of cables leading back to a hub or switch. A switch keeps track of which computers are connected to each port. When packets come in for a computer, the switch sends the packet back out on the port connecting to that computer and only on that port. (The only exception is broadcast packets, which are general packets addressed to everyone. These are, of course, sent out on all ports.) Now when you send any kind of packets over a hub, on the other hand, the hub sends those packets out every other port on the hub. On a hub, all traffic to *anybody* is repeated to *everybody*.

So why don't users connected to a hub see everyone's communications? Normally, a network card on a computer will automatically disregard any packets addressed to other computers. It listens for packets addressed to itself or to the broadcast address, and does not "accept delivery" of other packets. One of the capabilities that winpcap adds to a Windows PC is the ability to put the network card in promiscuous mode. In this mode, the card will accept all the traffic it sees, whatever the address on the packets. A sniffer program such as Windump can have the network card pass through all those packets for examination. In a switched environment, a network card in promiscuous mode would still see only packets meant for its own computer – a switch will only send traffic your way if it belongs to you. But on a hub, with a network card in promiscuous mode, you can see all the network traffic for all computers attached to the same hub.

The default mode for windump is to collect all traffic. However, this does not seem to be what we see in the capture file. All the packets in the capture file

are those that would have been received by Leila's computer – they are either broadcast packets or packets specifically addressed from or to her computer.

To use windump to capture traffic addressed to and from a single computer you must use the *host* argument as in:

h
s

```
C:\> windump -i 2 -w capture host LinuxForensics
```

T
i
C

ommand will capture from interface number 2 (-i 2) all packets addressed to or from the computer called LinuxForensics and save the packets to a file called capture. But there is a problem. This command cannot create the capture file on the USB drive.

With every network sniffer that I have used, using such a switch filters the traffic and discards every packet unless the source or destination address matches LinuxForensics' address. The command does not actually catch ALL of the traffic that LinuxForensics' NIC will accept. LinuxForensics will also accept broadcast packets. On broadcast packets, the source address will be the address of the originating device. The destination address will be either an IP address in the form x.x.x. 255 or a hardware address, ff:ff:ff:ff:ff:ff not the specific address of the receiving computer. So windump, if it acts like every other sniffer, would catch broadcast packets originating from LinuxForensics, but not any that are going to LinuxForensics. Broadcast packets originating with LinuxForensics would have its address in the source address field. Broadcast packets originating somewhere else but “heard” by LinuxForensics will not have the target address in either field and should not show up. In other words, the first six packets shown in the illustration above *should not be there*.

To verify this, I set up a test network consisting of two computers attached to a hub. On one computer, I installed winpcap and downloaded Windump. I ran Windump with the arguments listed above. On the second, LinuxForensics, I opened a web browser and opened a few pages. After a few minutes I had captured some three thousand packets. I opened the capture file in Ethereal. I sorted the packets by address and found, as I expected, that all the packets had LinuxForensics' address in either the source or the destination address. There were broadcast packets originating from LinuxForensics, but none that LinuxForensics was receiving from other devices.

Of course, there are ways to produce the kind of capture file seen above. One way is to use a more complex command: “c:\>windump -i 2 host LinuxForensics or dst broadcast”

And there are a number of other, still more complicated ways to achieve the same result. But there is no reason for an eavesdropper to go to the effort to

add this extra complexity. If the eavesdropper is experienced, he would know that there is nothing of interest in these extra packets. If he/she is inexperienced, it would require extra research and probably some trial and error to get it right. And remember, we are talking about a suspect who was flustered enough to make no effort to effectively wipe the deleted files. Yet this same person, at almost at the same moment, has the presence to correctly add unnecessary switches to the command line? It doesn't seem to add up.

At this point, another question starts to bother me. My test capture of web surfing on the LinuxForensics laptop produced over three thousand packets in just a few minutes. The capture file from the USB drive has only one hundred and thirteen packets. As capture files go, this one is *very* small. It is a remarkable coincidence that Robert Lawrence would find anything of interest in such a small capture. Another small problem? Maybe, maybe not. How accurately could Robert Lawrence gage the moment at which Leila would start sending e-mail as opposed to performing other activities on her computer? In fact, the time stamps on the packets in the capture file confirm that this file represents a capture lasting *less than one second!*

The SNMP packets present, to my mind, another serious problem. It is not so much that it would be odd for someone wiretapping another person's computer to have made the effort to capture these packets. It is a much bigger problem that they were created and sent out in the first place. SNMP is a protocol designed to remotely monitor and control network devices, such as routers and switches. An SNMP trap is an event occurring on a monitored device that triggers it to send a notification – an SNMP trap packet – usually to a network control application or to a logging server. The important point is that SNMP traps don't just happen. They must be configured in the device, and part of that configuration is telling the device where to send the trap packets. While it might be technically possible to configure a device to send traps to the broadcast address, it simply isn't done. You don't want these packets going to just anybody – only to your Network Operations Center. These trap packets often contain information that would be useful to a hacker. If you go to the trouble of configuring SNMP traps, you make sure to send them to the correct address. Yet you can see that these packets are addressed to 192.168.2.255, broadcast to *everybody* on the 192.168.2.x subnet. Simple carelessness or overwork or any explanation I can imagine does not account for the existence of these packets.

Another interesting thing we notice about these packets is that they seem to appear right after the computer makes a connection to a new outside address. And part of the content of the packet is this computer's address and the address of the server it connects to. This sort of information might be logged if this device were performing Network Address Translation (and we know that some device on this network was performing NAT). But, of course, there still would be no conceivable reason why we would instruct the router to send these

packets to the broadcast address, and some very good reasons why we would never send them there.

Another problem with these packets is that the read community string is “public.” The read community string is a sort of password used by SNMP indicating that the requestor is allowed to read information from the device. “Public” is the default read string on many devices. This string should have been replaced with a more secure string, but this precaution is sometimes neglected. SNMP managed devices will also have a write community string that allows the requestor to write to the device – in effect to issue commands on the device.

All these packets contain the string “enterprises.3955.1.1.0.” We can Google this string and hope we get some hits. Eureka! I was simply hoping to find out what vendor was identified by this string. The vendor is Linksys. But we got more than that. There are several online discussions about this string dating as far back as 2002. Apparently, some Linksys Cable/DSL routers had a bug in the firmware. Linksys apparently did not respond to any inquiries about the problem, but some enterprising customers did some tests of their own.³

Sending an SNMP query to one of these routers using read string “public” causes it to error, sending out lots of SNMP packets to the broadcast address on every interface. Worse still, once the router was sending out these packets, it was possible to *write* to the device even if it had been configured to accept no write commands from that interface (a common security precaution). In effect, with practice, someone could learn to take control of one of these routers and reconfigure it. The packets that these routers would send out in response to this bug looked very much like our six extra packets from the capture file.

Eventually, Linksys came out with a firmware upgrade that fixed this problem on the WAN interface – the interface that would face the outside world. But I was unable to discover if this problem still exists on the inside interface(s). At any rate, it is not at all uncommon for firmware in such devices to go years without an upgrade. Linksys is a brand aimed at the SOHO – Small Office Home Office – market. Usually these are for home use or for small businesses or small branch offices. In this situation there is rarely a staff person with the responsibility to keep the network infrastructure maintained. The device is set up and forgotten until the users report a problem.

Can we show that the router in our office was a Linksys? In a word, yes. The SNMP packets showed a source MAC address of 00:0c:41:50:29:2c. The first half of a MAC address is a vendor code. The vendor code in this case belongs to Linksys. The presence of a Linksys Cable/DSL router suggests that this is a small branch separate from the main company offices. So even two years after a patch was released, it is conceivable that it had not been applied.

Recommendations, Round 2

So what do we have? Six packets more than we wanted. The six SNMP packets indicate that the router is broadcasting information to everybody in the sales office and very likely to a large segment of public addresses out its wan interface. These broadcasts are most likely the result of someone sending an SNMP query to the router that activated a serious security vulnerability.

The company should immediately investigate its records, firewall, router and server logs to discover, if possible, how long the router has been in this state and whether this vulnerability has been leveraged to gain access to any sensitive corporate data. It should also audit security procedures. Specifically, the vulnerability in this router has been known for over two years. Why was no fix or workaround employed? Why didn't network services know that one of their routers was broadcasting SNMP traps?

This investigation should include logs of all VPN links to other vendors and customers. Were any sales associates making connections to partners that are unusual? Criminals are well aware that businesses usually give a number of vendors and customers tunnels to access necessary resources within the secure corporate network. A criminal will look for weak links in the networks of major vendors and customers of the company they are targeting. If these packets represent the presence of a determined hacker, he may actually be after one of our vendors or customers.

The case of Robert Lawrence and Leila Conlay must also be handled. At the moment, the digital evidence is inconclusive. Both employees should be placed on administrative leave immediately and all access to corporate information systems suspended while the wiretapping investigation proceeds. In many corporations, this is standard procedure when a claim of sexual harassment has been lodged.

For now, neither Robert nor Leila needs to know that anything more than sexual harassment is being investigated. It is very possible that a person who can install and run a packet sniffer with apparently little or no difficulty also has the skills to eliminate or obfuscate digital evidence. This must be prevented. Both of their PC's should be thoroughly and carefully examined for evidence that winpcap was installed as well as for other inappropriate activity.

Why do I suggest both computers be searched? Well, at the outset, this looked like a very simple case. Robert Lawrence allowed infatuation to become obsession and allowed obsession to lead him over the line that separates acceptable behavior from unacceptable behavior both socially and legally.

But even at the beginning there were little problems with this theory. Enough little problems have added up to merit considering other explanations

for the evidence. Here are some of the problems:

I am not a psychologist, but a progression from acting cordial to defensive to threatening within the course of four days seems unlikely. Isn't this awfully fast? Not a big problem in my mind, but still a problem.

Why wasn't the USB drive wiped clean of evidence? Someone who can sniff packets is almost certainly aware how easy it is to recover deleted files from a FAT16 partition. Wiping the drive or simply throwing it out would have been effective and simple methods to keep these files from being discovered.

How did our wiretapper manage to capture an e-mail of interest when he/she only captured one hundred and thirteen packets? This is extremely unlikely.

Yes, this could be part of a larger capture that was filtered. In fact, it probably was. But then the question becomes, why were these captures stored somewhere else and only the small portion of capture traffic that incriminated Robert Lawrence copied to the USB drive?

Why were the SNMP packets present in the capture file? If these packets were intercepted via remote sniffing, these represent still another instance where Robert Lawrence went to more work and more complexity than was necessary for no apparent reason.

Why were the Word documents created in the first place? It would have been so much simpler and quicker to create these short, simple messages within the e-mail program itself. And why copy the documents to the USB drive?

This is a whole series of actions which are of no apparent benefit to the sender of the e-mails. The only effect these actions have that differs from the effect of writing the e-mails in the normal manner is that they deposited evidence that incriminates Robert Lawrence on the USB drive.

Why were these files, and *only* these files, placed on the USB drive in the first place. Why not download winpcap only onto the hard drive of the computer on which it would be installed? The same with windump? Why copy the capture file to the USB drive? Why copy the map.gif file? It is a very simple map. It was hardly likely that a person would need a copy of the map to find this coffee shop. But if they did need it, the natural action would be to print a copy directly from the web page. Again, saving this file to the USB drive requires extra steps from Robert Lawrence with no apparent benefit.

However, I must admit that none of these objections, nor the sum of them, entirely *disproves* the theory we started with – that all these files were copied to this drive by Robert Lawrence. However, there is a possible scenario which does not suffer any difficulty from these anomalies. In fact, it predicts them.

Suppose that Leila Conlay has some reason to wish Robert Lawrence fired from his job. She could have orchestrated the whole chain of events. It would be simple to adjust her copy of MS Word to show Robert Lawrence as the registered user. She could have created the Word documents. She could have sniffed her own e-mail to Sam. She could have invited Robert to coffee Thursday evening in terms that would guarantee at least an uncomfortable

situation when she was already there with someone else. She could have saved all the pertinent files to a USB drive and placed the drive in Robert's cubicle before contacting corporate security.

How does this theory fit the evidence? Well all of the actions that were problems if the files were placed there by Robert Lawrence are completely compatible with this theory, and in fact are necessary for its success:

- The short timeline is not a problem.
- You might expect files on the USB drive to be deleted, but care would be taken to make sure at least most of them are recoverable. Therefore no unnecessary files would be added, only those that incriminate Robert.
- The capture file has been edited down to contain less than one second of activity. Once he had found the interesting packets, Robert would have had no reason to save this portion of the traffic as a separate file in a separate location. Leila, however, would want an investigator to find the interesting packets easily. She would also want to edit out as many unnecessary packets as possible, just in case any of them are inconsistent with the theory that Robert sniffed them.
- If Leila sniffed her own traffic, the SNMP packets would be captured unless extra command arguments were added to exclude them. Robert would not catch them unless he added extra arguments.
- The “e-mails” would show up as Word documents if Leila were manufacturing evidence. In this case, creating Word documents that implicate Robert is much easier than spoofing his return address on an e-mail.
- Of course, you would save all the incriminating files to a place where you could plant them as evidence. You would not have any reason to save any extra files to the USB drive. In fact, you would avoid it for fear of overwriting deleted files meant to be discovered or accidentally saving a file that points to yourself as the owner.

Ockham's razor says that, other things being equal, the simpler solution to a problem will be the correct solution. We have several instances in which the simpler solution is that Leila created this “evidence.”

The evidence fits this theory well enough that it ought to be investigated, if only to rule it out. The company has a stake in identifying the culprit correctly. This is a person with hacking skills, no scruples regarding lawbreaking and access to sensitive company data. The potential to harm or embarrass the company is significant.

The problem with the router may or may not be related to the twisted relationship of our two sales associates. It may represent an

automated scan which was never followed up, a full-fledged attack that may not even be aimed at CC Terminals, or just a faulty router. The attacker might be internal or external. But it has the potential to be much more damaging to the company than the sexual harassment complaint. It should be investigated diligently. If there are router, firewall or ids logs for the branch office, they should be examined for the original attack against the router, unusual connections to the sales office and outside connections that correspond chronologically to connections opened to corporate headquarters or partners.

In the present security climate, a Linksys router simply is not an acceptable choice for a branch office of a corporation. It should be replaced with a device that has more robust security and management features.

The legal implications of this case may differ depending on who is the culprit. But for practical purposes the actions of the company will almost certainly be the same. The infractions by either Robert or Leila do not justify the expense to the company and the possible publicity that would result from prosecution. The offending employee will be terminated. Under Nevada law, no cause need be given, nor should be given. The employee's file will simply be marked "no rehire." Once this is done, there is no danger of the company being embroiled in an action over hostile work environment since the offending employee was fired. The threat of prosecution under the wiretap statute, 18 U.S.C. 2511, can be held to discourage any acts of retaliation by the fired employee against the company.

Further investigation of the possible router hack may uncover actions the company would wish to prosecute. That would be a new case to which our only contribution would be six SNMP packets.

References

- 1) RFC 1321: <http://rfc.net/rfc1321.html>
- 2) Brian Carrier: Sleuthkit-users digest, Vol 1 #262, Feb 28, 2005
- 3) www.winpcap.polito.it
- 4) <http://www.governmentsecurity.org/articles/LinksysRouterInformationAcollection.php>

For Further Information

A discussion of what activity is considered sexual harassment and what companies can do to protect themselves can be found at many places on the Internet, including:

<http://www.de.psu.edu/harassment/>

An easy to follow explanation of the uses and mechanics of SNMP is at:

http://www.cuddletech.com/snmp_guide.pdf

Most of the tools used in this investigation come from the Sleuthkit. The Sleuthkit home site is:

<http://www.sleuthkit.org/>

Appendix A Contents of Timeline.all

```

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          0 .a. -rwxrwxrwx 0      0      12
<usbpart.img-_INDUMP.EXE-dead-12>
                                450560 .a. -/-rwxrwxrwx 0      0
12      /WinDump.exe (_INDUMP.EXE) (deleted)
                                485810 .a. -/-rwxrwxrwx 0      0
7      /WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
                                0 .a. -rwxrwxrwx 0      0      7
<usbpart.img-_INPCA~1.EXE-dead-7>

```

```

Wed Oct 27 2004 16:23:50 485810 m.. -/-rwxrwxrwx 0 0
10 /WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
485810 m.. -rwxrwxrwx 0 0 10
<usbpart.img-_INPCA~1.EXE-dead-10>
Wed Oct 27 2004 16:23:54 0 ..c -rwxrwxrwx 0 0 7
<usbpart.img-_INPCA~1.EXE-dead-7>
485810 ..c -/-rwxrwxrwx 0 0
7 /WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
485810 ..c -rwxrwxrwx 0 0 10
<usbpart.img-_INPCA~1.EXE-dead-10>
485810 ..c -/-rwxrwxrwx 0 0
10 /WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
Wed Oct 27 2004 16:23:56 0 m.. -rwxrwxrwx 0 0 7
<usbpart.img-_INPCA~1.EXE-dead-7>
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)
450560 m.. -rwxrwxrwx 0 0 14
<usbpart.img-_INDUMP.EXE-dead-14>
Wed Oct 27 2004 16:24:04 450560 ..c -/-rwxrwxrwx 0 0
14 /WinDump.exe (_INDUMP.EXE) (deleted)
450560 ..c -rwxrwxrwx 0 0 14
<usbpart.img-_INDUMP.EXE-dead-14>
450560 ..c -/-rwxrwxrwx 0 0
12 /WinDump.exe (_INDUMP.EXE) (deleted)
0 ..c -rwxrwxrwx 0 0 12
<usbpart.img-_INDUMP.EXE-dead-12>
Wed Oct 27 2004 16:24:06 0 m.. -rwxrwxrwx 0 0 12
<usbpart.img-_INDUMP.EXE-dead-12>
450560 m.. -/-rwxrwxrwx 0 0
12 /WinDump.exe (_INDUMP.EXE) (deleted)
Thu Oct 28 2004 00:00:00 53056 .a. -rwxrwxrwx 0 0 15
<usbpart.img-_apture-dead-15>
8814 .a. -/-rwxrwxrwx 0 0
16 /_ap.gif (deleted)
53056 .a. -/-rwxrwxrwx 0 0
15 /_apture (deleted)
450560 .a. -rwxrwxrwx 0 0 14
<usbpart.img-_INDUMP.EXE-dead-14>
8814 .a. -/-rwxrwxrwx 0 0
17 /_ap.gif (deleted)
19968 .a. -/-rwxrwxrwx 0 0
18 /coffee.doc
485810 .a. -rwxrwxrwx 0 0 10
<usbpart.img-_INPCA~1.EXE-dead-10>
450560 .a. -/-rwxrwxrwx 0 0
14 /WinDump.exe (_INDUMP.EXE) (deleted)
0 .a. -rwxrwxrwx 0 0 16
<usbpart.img-_ap.gif-dead-16>
485810 .a. -/-rwxrwxrwx 0 0
10 /WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
8814 .a. -rwxrwxrwx 0 0 17
<usbpart.img-_ap.gif-dead-17>
Thu Oct 28 2004 11:08:24 53056 ..c -rwxrwxrwx 0 0 15
<usbpart.img-_apture-dead-15>
53056 ..c -/-rwxrwxrwx 0 0
15 /_apture (deleted)

```



```

Thu Oct 28 2004 11:11:00    53056 m.. -/-rwxrwxrwx 0      0
15      /_apture (deleted)    53056 m.. -rwxrwxrwx 0      0      15
<usbpart.img-_apture-dead-15>
Thu Oct 28 2004 11:17:44      0 ..c -rwxrwxrwx 0      0      16
<usbpart.img-_ap.gif-dead-16>
8814 ..c -/-rwxrwxrwx 0      0
17      /_ap.gif (deleted)    8814 ..c -rwxrwxrwx 0      0      17
<usbpart.img-_ap.gif-dead-17>
8814 ..c -/-rwxrwxrwx 0      0
16      /_ap.gif (deleted)
Thu Oct 28 2004 11:17:46    8814 m.. -rwxrwxrwx 0      0      17
<usbpart.img-_ap.gif-dead-17>
0 m.. -rwxrwxrwx 0      0      16
<usbpart.img-_ap.gif-dead-16>
8814 m.. -/-rwxrwxrwx 0      0
17      /_ap.gif (deleted)    8814 m.. -/-rwxrwxrwx 0      0
16      /_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

```

© SANS Institute 2000 - 2005, Author retains full rights.