



# 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>

**Roger Hiew**

**Date Submitted: February 28, 2005**

**Analysis of an unknown USB JumpDrive image**

**GIAC Certified Forensic Analyst (GCFA)  
Practical Assignment Version 2.0 (Nov 18 2004)**

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

## **Abstract**

This practical assignment completes one of the requirements for the SANS GIAC Certified Forensic Analyst (GCFA) certification. The assignment is primarily divided into three sections.

The first section details the forensic investigation process used to determine the contents on the USB JumpDrive. The process includes determining the files on the USB JumpDrive, recovering files that were deleted on the USB JumpDrive, and finally analyzing the purpose of these files.

The second section describes the legal issues based on information discovered from this investigation. The section provides the Canada Laws that have been violated.

The third section provides recommendations as to what should occur after this investigation.

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

## Notation

Many of the forensic analysis in this assignment involve Linux commands. All of these commands are presented in 11 point Courier font, using the following format:

```
[root@LinuxForensics gcfa]# command arg ... arg  
result
```

Where:

- `root` is the current user.
- `LinuxForensics` is the name of the Forensics workstation.
- `gcfa` is the folder where the analysis is being done.
- `command` is the Linux command.
- `arg ... arg` are the list of arguments or options that are passed to the command.
- `result` is the information that is returned after the command has been run.

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

## Acronyms

USB	Universal Serial Bus
FAT	File Allocation Table
MD5	Message Digest 5
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
MAC	Modified, Accessed, Changed
NTFS	New Technology File System
UFS	Unix File System
DOS	Disk Operating System
GIF	Graphic Interchange Format
GUI	Graphical User Interface
SYN	Synchronize
ACK	Acknowledge
FIN	Finish
EOF	End of File
ICMP	Internet Control Message Protocol
PING	Packet Internet Groper
DLL	Dynamic Link Libraries
API	Application Programming Interface
ADSI	Active Directory Services Interfaces
COM	Component

SDK

Software Development Kit

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

## Table of Contents

<a href="#">Abstract</a>	2
<a href="#">Notation</a>	3
<a href="#">Acronyms</a>	4
<a href="#">List of Figures</a>	6
<a href="#">List of Tables</a>	8
<a href="#">Analysis of an unknown USB JumpDrive Image</a>	9
<a href="#">Introduction</a>	9
<a href="#">Executive Summary</a>	9
<a href="#">Examination Details</a>	10
<a href="#">Validate and Uncompress USB JumpDrive Image</a>	10
<a href="#">Extracting Partitions</a>	12
<a href="#">Forensic analysis on the Primary Table partition</a>	14
<a href="#">Forensic analysis on the unallocated partition</a>	15
<a href="#">Examination of the DOS FAT16 partition</a>	16
<a href="#">Recovery and Analysis of the deleted files</a>	19
<a href="#">Windump.exe</a>	19
<a href="#">_ap.gif</a>	24
<a href="#">_apture</a>	29
<a href="#">WinPCap_3_1_beta_3.exe</a>	34
<a href="#">Conclusions</a>	47
<a href="#">Image Details</a>	48
<a href="#">Forensic Details</a>	52
<a href="#">WinPcap_3_1_beta_3.exe Program Analysis</a>	54
<a href="#">WinDump.exe Program Analysis</a>	57
<a href="#">Program Identification</a>	66
<a href="#">Legal Implications</a>	71
<a href="#">Recommendations</a>	74
<a href="#">Additional Information</a>	77
<a href="#">List of References</a>	78

## List of Figures

<a href="#"><u>Figure 1 – Uncompress the USB JumpDrive Image using unzip</u></a>	11
<a href="#"><u>Figure 2 – File type classification of the USB JumpDrive Image</u></a>	11
<a href="#"><u>Figure 3 – Rename USB JumpDrive Image file extension</u></a>	11
<a href="#"><u>Figure 4 – MD5 hash of the USB JumpDrive Image</u></a>	12
<a href="#"><u>Figure 5 – File type classification on the USB JumpDrive Image</u></a>	12
<a href="#"><u>Figure 6 – Partitions on USB JumpDrive image</u></a>	12
<a href="#"><u>Figure 7 – Extracting the primary table partition</u></a>	13
<a href="#"><u>Figure 8 – Extracting the unallocated partition</u></a>	14
<a href="#"><u>Figure 9 – Extracting the FAT16 partition</u></a>	14
<a href="#"><u>Figure 10 – File type classification on the primary table partition</u></a>	14
<a href="#"><u>Figure 11 – Strings output for the primary table partition</u></a>	15
<a href="#"><u>Figure 12 – Khxedit of the unallocated partition</u></a>	16
<a href="#"><u>Figure 13 – FIs and Mactime of the FAT16 partition</u></a>	17
<a href="#"><u>Figure 14 – Timeline of the FAT16 image</u></a>	18
<a href="#"><u>Figure 15 – Fsstat of the FAT16 image</u></a>	20
<a href="#"><u>Figure 16 – Istat on inode address 12 of the FAT16 image</u></a>	21
<a href="#"><u>Figure 17 – Istat on inode address 12 of the FAT16 image</u></a>	22
<a href="#"><u>Figure 18 – Dcfldd of the WinDump.exe file</u></a>	23
<a href="#"><u>Figure 19 – File type classification of the WinDump.exe file</u></a>	23
<a href="#"><u>Figure 20 – MD5 hash of the downloaded WinDump.exe file</u></a>	24
<a href="#"><u>Figure 21 – Istat on inode address 16 of the FAT16 image</u></a>	24
<a href="#"><u>Figure 22 – Istat on inode address 17 of the FAT16 image</u></a>	25
<a href="#"><u>Figure 23 – Dcfldd of the ap.gif file</u></a>	25
<a href="#"><u>Figure 24 – Khxedit of the recovered ap.gif file</u></a>	26
<a href="#"><u>Figure 25 – Khxedit of the modified ap.gif file</u></a>	27
<a href="#"><u>Figure 26 – File type classification on the ap.gif file</u></a>	28
<a href="#"><u>Figure 27 – The Map</u></a>	28
<a href="#"><u>Figure 28 – Rename of the ap.gif file to map.gif</u></a>	28
<a href="#"><u>Figure 29 – Istat on inode address 15 of the FAT16 image</u></a>	29
<a href="#"><u>Figure 30 – Dcfldd of the apture file</u></a>	30
<a href="#"><u>Figure 31 – File type classification of the apture file</u></a>	30
<a href="#"><u>Figure 32 – Rename of the apture file to capture</u></a>	31
<a href="#"><u>Figure 33 – Ethereal of the capture file</u></a>	31
<a href="#"><u>Figure 34 – The TCP Data Stream</u></a>	33
<a href="#"><u>Figure 35 – Istat on inode address 10 of the FAT16 image</u></a>	35
<a href="#"><u>Figure 36 – Dcfldd of the WinPcap 3 1 beta 3.exe file</u></a>	36
<a href="#"><u>Figure 37 – MD5 hash of the downloaded WinPcap 3 1 beta 3.exe file</u></a>	36
<a href="#"><u>Figure 38 – Khxedit of the partially recovered WinPcap 3 1 beta 3.exe file</u></a>	37
<a href="#"><u>Figure 39 – Khxedit of the downloaded WinPcap 3 1 beta 3.exe file</u></a>	38
<a href="#"><u>Figure 40 – Khxedit of the “pasted” WinPcap 3 1 beta3.exe file</u></a>	39
<a href="#"><u>Figure 41 – The end of file (EOF) of the downloaded WinPcap 3 1 beta 3.exe file</u></a>	40
<a href="#"><u>Figure 42 – The end of file (EOF) of the recovered WinPcap 3 1 beta 3.exe file #1</u></a>	41
<a href="#"><u>Figure 43 – The end of file (EOF) of the recovered WinPcap 3 1 beta 3.exe file #2</u></a>	42



<a href="#"><u>Figure 44 – The complete recovered WinPcap 3 1 beta 3.exe file</u></a>	43
<a href="#"><u>Figure 45 – MD5 hash of the complete recovered WinPcap 3 1 beta 3.exe file</u></a>	44
<a href="#"><u>Figure 46 – Mounting the FAT16 image</u></a>	44
<a href="#"><u>Figure 47 – Files on the FAT16 image</u></a>	45
<a href="#"><u>Figure 48 – File type classification of the files on the FAT16 image</u></a>	45
<a href="#"><u>Figure 49 – The her.doc document</u></a>	45
<a href="#"><u>Figure 50 – The properties of the her.doc document</u></a>	46
<a href="#"><u>Figure 51 – The hey.doc document</u></a>	46
<a href="#"><u>Figure 52 – The coffee.doc document</u></a>	47
<a href="#"><u>Figure 53 – File listing of the FAT16 image</u></a>	48
<a href="#"><u>Figure 54 – File listing of the recovered files</u></a>	48
<a href="#"><u>Figure 55 – The MD5 hashes of the her.doc file</u></a>	48
<a href="#"><u>Figure 56 – The MD5 hashes of the hey.doc file</u></a>	48
<a href="#"><u>Figure 57 – The MD5 hashes of the coffee.doc file</u></a>	49
<a href="#"><u>Figure 58 – The MD5 hashes of the WinDump.exe file</u></a>	49
<a href="#"><u>Figure 59 – The MD5 hashes of the map.gif file</u></a>	49
<a href="#"><u>Figure 60 – The MD5 hashes of the capture file</u></a>	49
<a href="#"><u>Figure 61 – The MD5 hashes of the WinPcap 3 1 beta 3.exe file</u></a>	49
<a href="#"><u>Figure 62 – Timeline of the FAT 16 image</u></a>	50
<a href="#"><u>Figure 63 – The dirty word list</u></a>	51
<a href="#"><u>Figure 64 – MD5 Hash of the WinDump and WinPcap programs</u></a>	54
<a href="#"><u>Figure 65 – Strace output for WinPcap installation</u></a>	54
<a href="#"><u>Figure 66 – Strace output “winpcap.txt “ file</u></a>	55
<a href="#"><u>Figure 67 – Strace output for WinDump capture</u></a>	57
<a href="#"><u>Figure 68 – Ping 192.168.2.1 (Gateway)</u></a>	58
<a href="#"><u>Figure 69 – MD5 hash of the windump-capture file</u></a>	58
<a href="#"><u>Figure 70 – MD5 verification of the windump-capture file</u></a>	58
<a href="#"><u>Figure 71 – Ethereal of the windump-capture file</u></a>	59
<a href="#"><u>Figure 72 – Khexedit of the windump-capture file #1</u></a>	62
<a href="#"><u>Figure 73 – Khexedit of the windump-capture file #2</u></a>	63
<a href="#"><u>Figure 74 – Compile the release version of WinDump</u></a>	67
<a href="#"><u>Figure 75 – Compiling WinDump (Release version)</u></a>	67
<a href="#"><u>Figure 76 – The compiled WinDump program (Release version)</u></a>	67
<a href="#"><u>Figure 77 – MD5 hash of the compiled WinDump program (Release version)</u></a>	68
<a href="#"><u>Figure 78 – Testing the compiled WinDump program (Release version)</u></a>	68
<a href="#"><u>Figure 79 – The WinDump capture file</u></a>	68

## List of Tables

<a href="#"><u>Table 1 – Deleted files</u></a>	19
<a href="#"><u>Table 2 – WinPcap.txt Strace Output #1</u></a>	55
<a href="#"><u>Table 3 – WinPcap.txt Strace Output #2</u></a>	56
<a href="#"><u>Table 4 – WinPcap.txt Strace Output #3</u></a>	56
<a href="#"><u>Table 5 – Strace output for WinDump program #1</u></a>	60
<a href="#"><u>Table 6 – Strace output for WinDump program #3</u></a>	61
<a href="#"><u>Table 7 – Strace output for WinDump program #4</u></a>	61
<a href="#"><u>Table 8 – Strace output for WinDump program #5</u></a>	62
<a href="#"><u>Table 9 – Strace output for WinDump program #6</u></a>	63
<a href="#"><u>Table 10 – Strace output for WinDump program #7</u></a>	64
<a href="#"><u>Table 11 – Strace output for WinDump program #8</u></a>	64
<a href="#"><u>Table 12 – Strace output for WinDump program #9</u></a>	65

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

# Analysis of an unknown USB JumpDrive Image

## Introduction

On October 29<sup>th</sup> 2004, Leila Conlay (an employee at CC Terminals) made an official complaint on Robert Lawrence (another employee) to the corporate security office. Ms. Conlay's complaint stated that Mr. Lawrence has been harassing her. He also made numerous attempts to meet her during and outside of work hours. Ms. Conlay also stated that Mr. Lawrence has contacted her through her personal email address. On the evening of October 28<sup>th</sup> 2004, Ms. Conlay was having coffee with a friend at a secluded location when Mr. Lawrence suddenly appeared. Upon receiving this complaint, the corporate security office ordered an investigation on Mr. Lawrence. After hours of search of Mr. Lawrence's cubicle, Mark Mawer (the security administrator at CC Terminals) found a USB JumpDrive. This USB JumpDrive was entered into evidence as follows:

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

The purpose of this practical assignment is to analyze this USB JumpDrive image and determine how Robert Lawrence had used it.

## Executive Summary

A forensic analysis of the USB JumpDrive found in Robert Lawrence's cubicle has revealed clear evidence to suggest that Robert Lawrence obtained Leila Conlay's personal information without her consent. This information was obtained by using a network wiretap program to intercept Leila Conlay's private email conversations.

Through the use of this network wiretap program, Mr. Lawrence was able to intercept an email conversation between Ms. Conlay and her friend. From this network wiretap, Mr. Lawrence was able to obtain Ms. Conlay's and her friend's personal email addresses. Mr. Lawrence was also able to obtain the time and location of the planned meeting between Ms. Conlay and her friend. Since Ms. Conlay did not give Mr. Lawrence the consent to wiretap, Mr. Lawrence through his actions has violated the Invasion of Privacy Law in Canada [33].

Forensic analysis tools were used to establish a timeline surrounding the actions of Mr. Lawrence. The timeline shows that Mr. Lawrence had deleted a total of four files on the USB JumpDrive. The deleted files include a network wiretap program, a program that contains libraries for the network wiretap program and the wiretap network capture file.

This wiretap network capture file contained the actual email conversation between Ms. Conlay and her friend. It is believed that Mr. Lawrence deleted these files in an attempt to conceal the fact that he had downloaded, installed and used a network wiretap program. Mr. Lawrence also deleted a fourth file. This fourth file contained a map of the meeting location between Ms. Conlay and her friend.

There were three other files on the USB JumpDrive. These files contained the email messages that were sent by Mr. Lawrence to Ms. Conlay and his emails were becoming more aggressive over time. The contents of these email messages revealed that Ms. Conlay repeatedly rejected Mr. Lawrence's attempts to meet with her outside of work hours. Mr. Lawrence however ignored Ms. Conlay's rejection and he continued to pursue her. At one point, Mr. Lawrence went so far as to threaten the possibility of harming Ms. Conlay. Through this action, Mr. Lawrence has violated the Canadian harassment laws, which state that it is an indictable offence if anyone knowingly utters a threat to cause bodily harm to another person. This offence is liable to imprisonment [35].

The results of this forensic analysis have produced sufficient evidence to warrant a legal action against Mr. Lawrence if this is the action Ms. Conlay wishes to pursue. Furthermore, it would be within the rights of CC Terminals to discipline Mr. Lawrence.

### **Examination Details**

The file GCFAPractical2.0-USBImageAndInfo.zip was downloaded from the SANS portal website. The cryptographic MD5 one-way hash of this image is 338ecf17b7fc85bbb2d5ae2bbc729dd5 (also obtained from the SANS portal website).

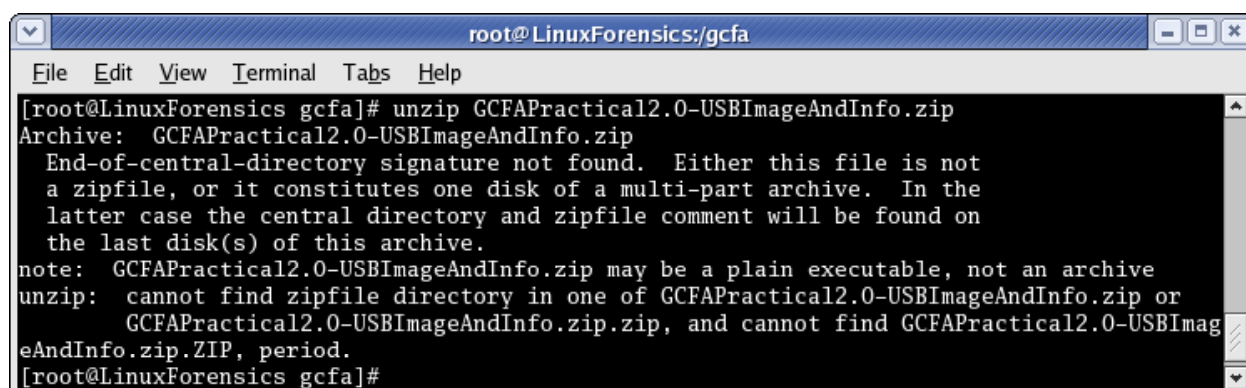
The MD5 hash of a file is essentially a digital fingerprint of the file. The MD5 hash is calculated from a complex mathematical algorithm and is 128-bit in length. The MD5 hash is also known as a one-way hash, which denotes that the calculated MD5 hash of a file can never be reversed to obtain the original data. Furthermore, "no two different files can create the same hash" [1]. The MD5 hash is important as it can be entered into evidence and can be used to prove if a file has been tampered with. Thus, the MD5 hash is often used to verify the integrity of files. In this forensic analysis, the **md5sum** utility was used to calculate the MD5 hash.

The forensic analysis machine used was an IBM S50 Thinkcentre desktop computer. The Redhat Fedora Core 2 Linux operating system was installed on this computer. The operating system was fully patched. The SANS Track 8 Linux forensic tools were installed on this computer. Since the content of the USB JumpDrive image was still unknown, the forensic machine was not connected to the network.

### **Validate and Uncompress USB JumpDrive Image**

The first step was to copy the downloaded file to the forensic machine via a Sandisk 512MB JumpDrive. The next step was to uncompress the file GCFAPractical2.0-

USBImageAndInfo.zip. The downloaded file was assumed to be a zip file because of the .zip file extension. Therefore, the **unzip** utility was used. This **unzip** utility will uncompress any file that has been compressed with the **zip** utility.




```
root@LinuxForensics:gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# unzip 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.
note:  GCFAPractical2.0-USBImageAndInfo.zip may be a plain executable, not an archive
unzip:  cannot find zipfile directory in one of GCFAPractical2.0-USBImageAndInfo.zip or
       GCFAPractical2.0-USBImageAndInfo.zip.zip, and cannot find GCFAPractical2.0-USBImag
eAndInfo.zip.ZIP, period.
[root@LinuxForensics gcfa]#
```

Figure 1 – Uncompress the USB JumpDrive Image using unzip

Figure 1 shows that the result of the unzip process. The result indicated that the downloaded file might be corrupted or it might not be a zip file, ie the file extension was labelled incorrectly.

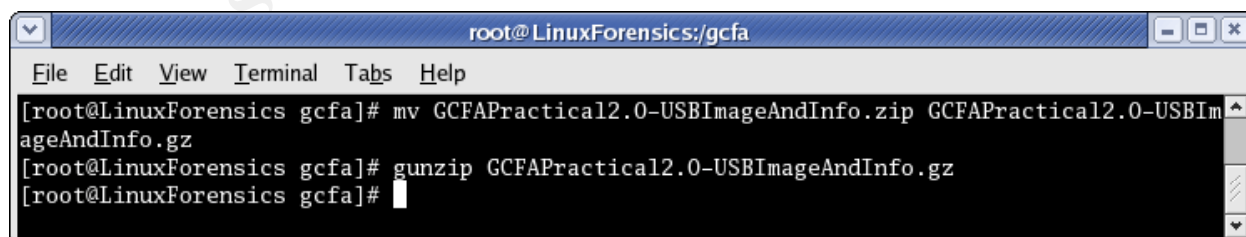
The **file** utility was used to examine this file further. The **file** utility determines the file type by looking at the file header and then comparing it to the list of known file types.



```
root@LinuxForensics:gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# file GCFAPractical2.0-USBImageAndInfo.zip
GCFAPractical2.0-USBImageAndInfo.zip: gzip compressed data, was "USBFD-64531026-RL-001.img", from Unix, max compression
[root@LinuxForensics gcfa]#
```

Figure 2 – File type classification of the USB JumpDrive Image

The **file** utility indicated that the downloaded file was a gzip file. The next step was to change the existing file extension to a gzip extension with the **mv** utility and use the **gunzip** utility to uncompress the downloaded file.

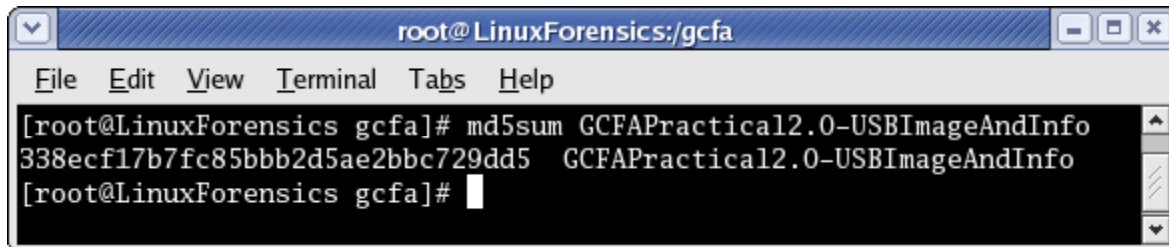


```
root@LinuxForensics:gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# mv GCFAPractical2.0-USBImageAndInfo.zip GCFAPractical2.0-USBImageAndInfo.gz
[root@LinuxForensics gcfa]# gunzip GCFAPractical2.0-USBImageAndInfo.gz
[root@LinuxForensics gcfa]#
```

Figure 3 – Rename USB JumpDrive Image file extension

The uncompressed file name was "**GCFAPractical2.0-USBImageAndInfo**". This was

the USB JumpDrive image file. The **md5sum** utility was used to obtain the MD5 hash of this uncompressed file. The MD5 hash was then compared to the one that was entered into evidence (338ecf17b7fc85bbb2d5ae2bbc729dd5).

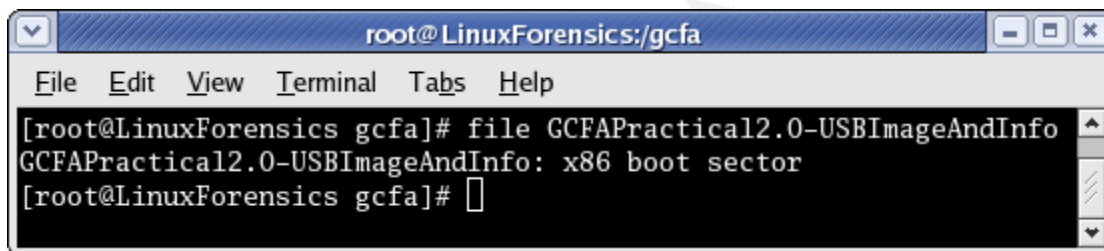
A terminal window titled 'root@LinuxForensics:/gcfa' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command 'md5sum GCFAPractical2.0-USBImageAndInfo' is entered, and the output '338ecf17b7fc85bbb2d5ae2bbc729dd5 GCFAPractical2.0-USBImageAndInfo' is displayed. The prompt is then '[root@LinuxForensics gcfa]# ' with a cursor.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# md5sum GCFAPractical2.0-USBImageAndInfo
338ecf17b7fc85bbb2d5ae2bbc729dd5 GCFAPractical2.0-USBImageAndInfo
[root@LinuxForensics gcfa]#
```

Figure 4 – MD5 hash of the USB JumpDrive Image

The MD5 hash was identical, and so it was confirmed that this was the correct image file to be examined. The next step was to use the **file** utility to determine the file type of this image file.

### Extracting Partitions

A terminal window titled 'root@LinuxForensics:/gcfa' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command 'file GCFAPractical2.0-USBImageAndInfo' is entered, and the output 'GCFAPractical2.0-USBImageAndInfo: x86 boot sector' is displayed. The prompt is then '[root@LinuxForensics gcfa]# ' with a cursor.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# file GCFAPractical2.0-USBImageAndInfo
GCFAPractical2.0-USBImageAndInfo: x86 boot sector
[root@LinuxForensics gcfa]#
```

Figure 5 – File type classification on the USB JumpDrive Image

The **file** command was then used, and indicated that this image file was formatted by a x86 machine. The **mmls** utility was used next.

The **mmls** utility reads an image file and returns the primary, extended and logical partitions that have been allocated to the image file. It also returns the location and size of these partitions. The **mmls** utility was run with the **-t dos** option. This option specifies the operating system used to create the partitions.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# mmls -t dos GCFAPractical2.0-USBIImageAndInfo
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)
[root@LinuxForensics gcfa]#
```

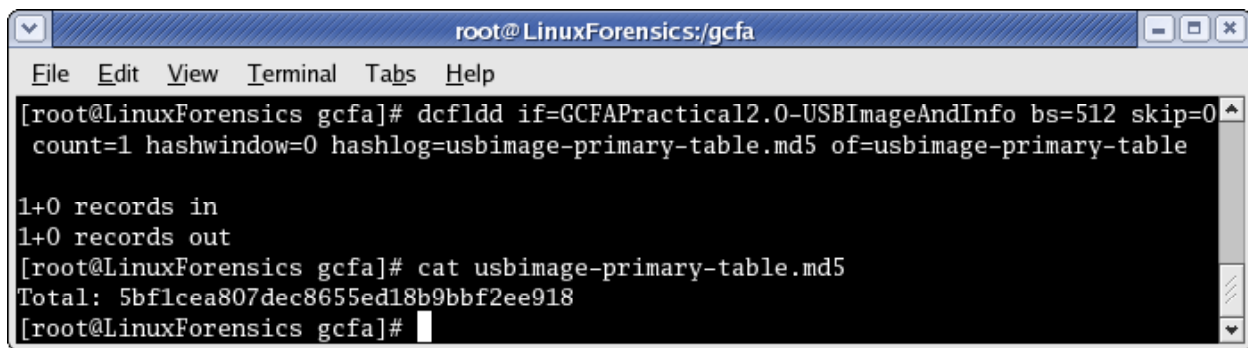
Figure 6 – Partitions on USB JumpDrive image

The information gathered from the **mmls** utility (Figure 6) indicated that there were three existing partitions: the Primary Table, an unallocated partition and a DOS FAT16 partition. All three partitions were extracted using the **dcfldd** utility for examination.

The **dcfldd** utility is used to create an image of a harddrive or partition. It can also be used to extract partitions or files from an image. The **dcfldd** utility is similar to the **dd** utility with the exception that the **dcfldd** utility provides a percentage completion status during the imaging process and the **dcfldd** utility can calculate the MD5 hash of the created or extracted image file.

To extract the partitions from the USB JumpDrive image, the **dcfldd** utility was run on each partition with 7 options:

- **if** – this option specifies the input image file, partition or harddrive devices that the **dcfldd** utility will be imaging.
- **bs** – this option specifies the block size.
- **skip** – this option specifies the starting point of the imaging process.
- **count** – this option specifies the length (in blocks) to copy.
- **hashwindow** – this option specifies the amount of data that must be processed before creating a MD5 hash. The **hashwindow=0** invokes the **dcfldd** utility to create the MD5 hash of the image once the imaging process has completed.
- **hashlog** – this option specifies the name of the file that the **dcfldd** utility will use to store the MD5 hash.
- **of** – this option specifies the output image file name.

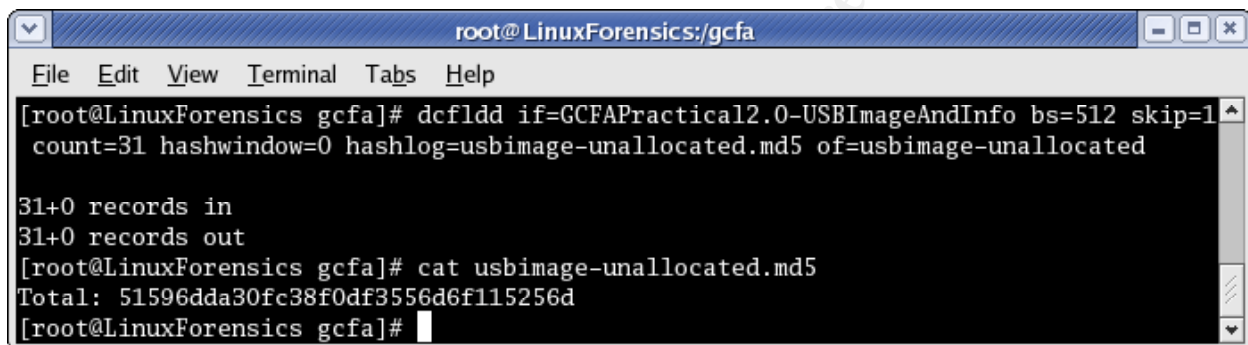
A terminal window titled 'root@LinuxForensics:/gcfa' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command 'dcflddd if=GCFAPractical2.0-USBIImageAndInfo bs=512 skip=0 count=1 hashwindow=0 hashlog=usbimage-primary-table.md5 of=usbimage-primary-table' is entered. The output shows '1+0 records in' and '1+0 records out'. Then, 'cat usbimage-primary-table.md5' is run, showing the MD5 hash 'Total: 5bf1cea807dec8655ed18b9bbf2ee918'.

```
root@LinuxForensics:/gcfa
[root@LinuxForensics gcfa]# dcflddd if=GCFAPractical2.0-USBIImageAndInfo bs=512 skip=0
count=1 hashwindow=0 hashlog=usbimage-primary-table.md5 of=usbimage-primary-table

1+0 records in
1+0 records out
[root@LinuxForensics gcfa]# cat usbimage-primary-table.md5
Total: 5bf1cea807dec8655ed18b9bbf2ee918
[root@LinuxForensics gcfa]#
```

Figure 7 – Extracting the primary table partition

Figure 7 shows the command that was used to extract the primary partition and the MD5 hash of the primary partition.

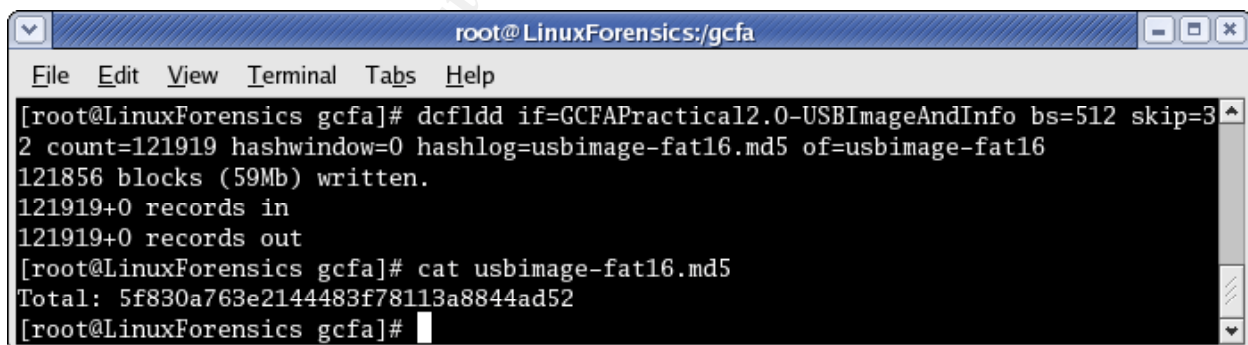
A terminal window titled 'root@LinuxForensics:/gcfa' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command 'dcflddd if=GCFAPractical2.0-USBIImageAndInfo bs=512 skip=1 count=31 hashwindow=0 hashlog=usbimage-unallocated.md5 of=usbimage-unallocated' is entered. The output shows '31+0 records in' and '31+0 records out'. Then, 'cat usbimage-unallocated.md5' is run, showing the MD5 hash 'Total: 51596dda30fc38f0df3556d6f115256d'.

```
root@LinuxForensics:/gcfa
[root@LinuxForensics gcfa]# dcflddd if=GCFAPractical2.0-USBIImageAndInfo bs=512 skip=1
count=31 hashwindow=0 hashlog=usbimage-unallocated.md5 of=usbimage-unallocated

31+0 records in
31+0 records out
[root@LinuxForensics gcfa]# cat usbimage-unallocated.md5
Total: 51596dda30fc38f0df3556d6f115256d
[root@LinuxForensics gcfa]#
```

Figure 8 – Extracting the unallocated partition

Figure 8 shows the command that was used to extract the unallocated partition and the MD5 hash of the unallocated partition.

A terminal window titled 'root@LinuxForensics:/gcfa' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command 'dcflddd if=GCFAPractical2.0-USBIImageAndInfo bs=512 skip=32 count=121919 hashwindow=0 hashlog=usbimage-fat16.md5 of=usbimage-fat16' is entered. The output shows '121856 blocks (59Mb) written.', '121919+0 records in', and '121919+0 records out'. Then, 'cat usbimage-fat16.md5' is run, showing the MD5 hash 'Total: 5f830a763e2144483f78113a8844ad52'.

```
root@LinuxForensics:/gcfa
[root@LinuxForensics gcfa]# dcflddd if=GCFAPractical2.0-USBIImageAndInfo bs=512 skip=3
2 count=121919 hashwindow=0 hashlog=usbimage-fat16.md5 of=usbimage-fat16

121856 blocks (59Mb) written.
121919+0 records in
121919+0 records out
[root@LinuxForensics gcfa]# cat usbimage-fat16.md5
Total: 5f830a763e2144483f78113a8844ad52
[root@LinuxForensics gcfa]#
```

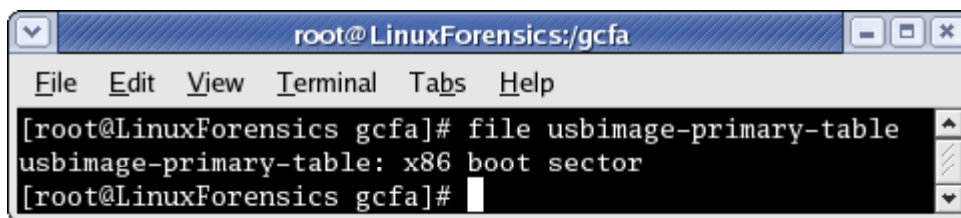
Figure 9 – Extracting the FAT16 partition

Figure 9 shows the command that was used to extract the FAT16 partition and the MD5 hash of the FAT16 partition.

### **Forensic analysis on the Primary Table partition**

The **file** utility was used to determine the file type of the primary table partition.



A terminal window titled 'root@LinuxForensics:/gcfa' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command 'file usbimage-primary-table' has been executed, resulting in the output 'usbimage-primary-table: x86 boot sector'.

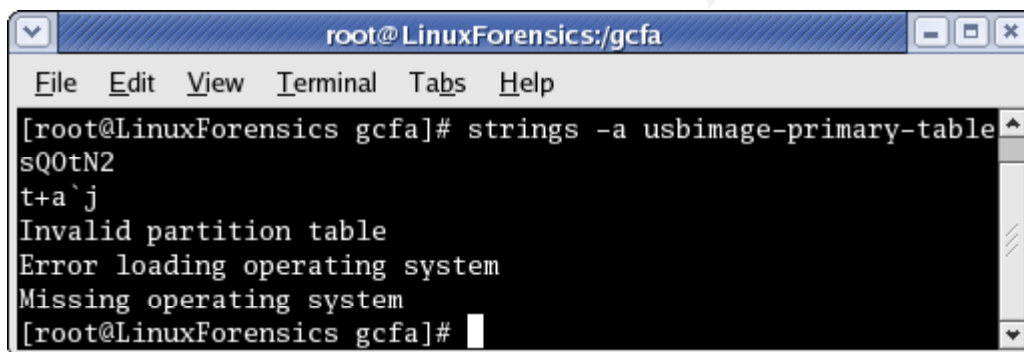
```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# file usbimage-primary-table
usbimage-primary-table: x86 boot sector
[root@LinuxForensics gcfa]#
```

Figure 10 – File type classification on the primary table partition

Figure 10 shows that this is the boot sector partition. “A file system boot sector is the first physical sector on a logical volume. In the case of hard drives, the first sector is referred to as the Master Boot Record and contains a partition table which describes the layout of the logical partition on that hard drive” [2].

The **strings** utility was used to examine the partition further. The **strings** utility parses through a file or image and outputs any readable string. If the file or image is unknown, the readable strings can provide hints as to what is on the image or file. This utility was run with the following option:

- **-a** – this option extracts all the readable strings in the file

A terminal window titled 'root@LinuxForensics:/gcfa' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command 'strings -a usbimage-primary-table' has been executed, resulting in the output: 'sQ0tN2', 't+a`j', 'Invalid partition table', 'Error loading operating system', and 'Missing operating system'.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# strings -a usbimage-primary-table
sQ0tN2
t+a`j
Invalid partition table
Error loading operating system
Missing operating system
[root@LinuxForensics gcfa]#
```

Figure 11 – Strings output for the primary table partition

Figure 11 shows the **strings** utility output. It was concluded that this partition was just the boot sector partition and that it did not contain anything worth examining further.

### Forensic analysis on the unallocated partition

This unallocated partition should contain no data since the **mmls** utility classified it as unallocated. The **khxedit** utility was used to confirm that this partition contained no information. The **khxedit** utility opens any file and displays the content of the file in hexadecimal format.

```
[root@LinuxForensics gcfa]# khxedit usbimage-unallocated
```

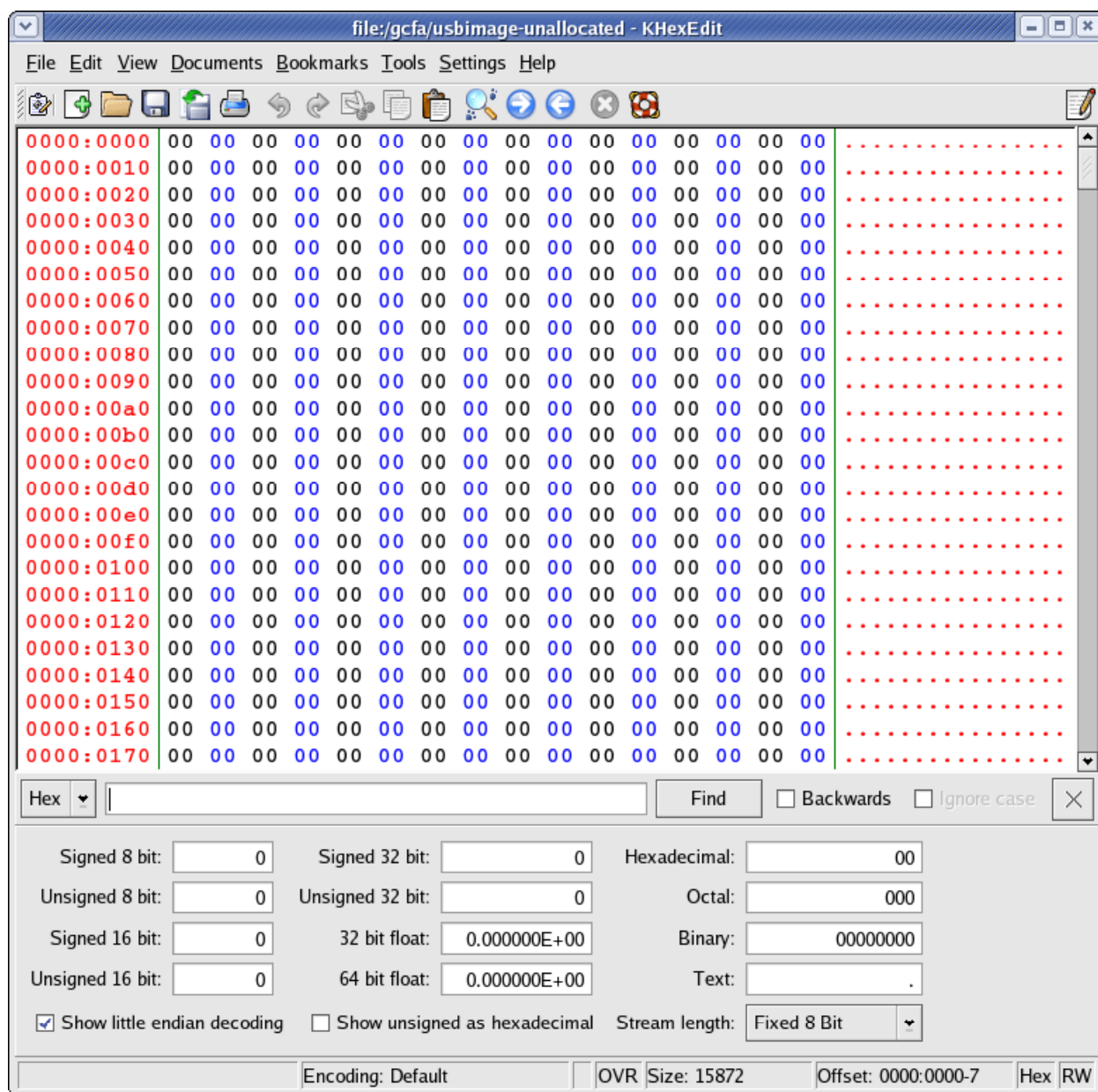


Figure 12 – Khexedit of the unallocated partition

Figure 12 shows that the unallocated partition was filled with all "00"s. The "00"s indicated that there was no data on this partition.

### Examination of the DOS FAT16 partition

According to the **mmls** utility, this is a DOS FAT16 partition and should contain data that will need to be examined. Therefore, the first step of this examination was to obtain the timeline.

Every file in the Windows or Unix file system has a last modified time, the last accessed time and the last time the content of the inode was changed. These

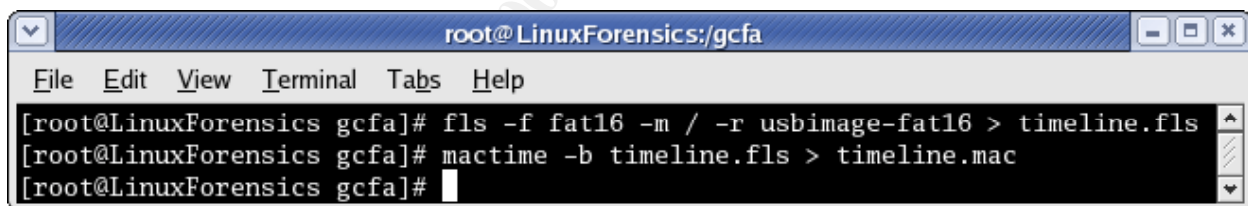
timestamps are also known as the MAC timeline – the modified, accessed and changed times). The inode of a file contains information such as the MAC times, the file size, the user ID and group ID of this file, the file type, the inode address and the permission of the file. The MAC timeline is extremely valuable as it provides an accurate representation of the events that have occurred in a file system. The MAC timeline is usually obtained first in order to avoid any accidental modification of any of the timestamps. For example, if a file is accessed accidentally, this will change the last accessed time and therefore, the timeline will not be accurate.

The **fls** utility can extract the MAC timeline of an image. In addition to that, it can also extract other inode information such as the inode numbers, the name of the file, the permission, the size, the owner and the group of all the files, including the deleted files. The **fls** utility was run with the following options (Figure 13):

- **-f** – this option specifies the image file system type (eg. FAT16, NTFS, Linux ext2, Linux ext2, Solaris UFS, etc).
- **-m /** – this option invokes the **fls** utility to obtain the MAC. The / indicates that the **fls** utility will start the operation at the root of the partition.
- **-r** – this option indicates that the **fls** utility will traverse through all the child folders.

The **mactime** utility takes the results from the **fls** utility and chronologically sorts the events based on the MAC time so that the timestamps are in readable format. The **mactime** utility was run with the following option (Figure 13):

- **-b** – this option indicates the **fls** timeline output file.

A screenshot of a terminal window titled 'root@LinuxForensics:/gcfa'. The terminal shows three commands being executed: 1. '[root@LinuxForensics gcfa]# fls -f fat16 -m / -r usbimage-fat16 > timeline.fls', 2. '[root@LinuxForensics gcfa]# mactime -b timeline.fls > timeline.mac', and 3. '[root@LinuxForensics gcfa]#'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Terminal', 'Tabs', and 'Help'. The background is black with white text. There are standard window control buttons (minimize, maximize, close) in the top right corner.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# fls -f fat16 -m / -r usbimage-fat16 > timeline.fls
[root@LinuxForensics gcfa]# mactime -b timeline.fls > timeline.mac
[root@LinuxForensics gcfa]#
```

Figure 13 – Fls and Mactime of the FAT16 partition

Figure 13 shows the sorted timeline was written to the file “timeline.mac”.

```

root@LinuxForensics:gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# cat timeline.mac
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 (_I
NPCA~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 (_I
NPCA~1.EXE) (deleted)
Wed Oct 27 2004 16:23:54 485810 ..c -/-rwxrwxrwx 0 0 7 /WinPcap_3_1_beta_3.exe (_I
NPCA~1.EXE) (deleted)
485810 ..c -/-rwxrwxrwx 0 0 10 /WinPcap_3_1_beta_3.exe (_I
NPCA~1.EXE) (deleted)
Wed Oct 27 2004 16:23:56 485810 m.. -/-rwxrwxrwx 0 0 7 /WinPcap_3_1_beta_3.exe (_I
NPCA~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 12 /WinDump.exe (_INDUMP.EXE)
(deleted)
450560 ..c -/-rwxrwxrwx 0 0 14 /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 8814 .a. -/-rwxrwxrwx 0 0 16 /_ap.gif (deleted)
485810 .a. -/-rwxrwxrwx 0 0 10 /WinPcap_3_1_beta_3.exe (_I
NPCA~1.EXE) (deleted)
53056 .a. -/-rwxrwxrwx 0 0 15 /_apture (deleted)
8814 .a. -/-rwxrwxrwx 0 0 17 /_ap.gif (deleted)
19968 .a. -/-rwxrwxrwx 0 0 18 /coffee.doc
450560 .a. -/-rwxrwxrwx 0 0 14 /WinDump.exe (_INDUMP.EXE)
(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
[root@LinuxForensics gcfa]#

```

Figure 14 – Timeline of the FAT16 image

Figure 14 shows the timeline extracted by the fls utility. The first 5 columns represented the time, in day of the week, month, date of the month, year, and time. The 6<sup>th</sup> column shows the file size. The 7<sup>th</sup> column shows the MAC flag to indicate which MAC times were associated with each file. The 8<sup>th</sup> column shows the file permission. The 9<sup>th</sup> and 10<sup>th</sup> columns show the user and the group IDs. The 11<sup>th</sup> column shows the inode address and the 12<sup>th</sup> column shows the file name. The file names appended with “(deleted)” in the 11<sup>th</sup> column signify that these files were deleted.

According to the timeline, there were 4 deleted files:

No.	Deleted File Name	Inode Addresses
1	WinDump.exe	12 and 14
2	_ap.gif	16 and 17
3	_apture	15
4	WinPcap_3_1_beta_3.exe	7 and 10

Table 1 – Deleted files

### **Recovery and Analysis of the deleted files**

In order to move forward in the investigation, the 4 deleted files (Table 1) had to be recovered. The content and program identification of these files would be used to provide hints as to why they existed on this partition and why they were deleted from this partition.

#### **Windump.exe**

The WinDump.exe file was recovered using the following procedures:

First, the **fsstat** utility was used. The **fsstat** utility provides information such as the file system type, file volume name, data content, sector size, etc on the image file. This information does vary between file systems, for example, the result of **fsstat** on a FAT16 file system is different from a Linux EXT3 file system. The **fsstat** utility was run with the following option:

- **-f** – this option specifies the image file system type (eg. FAT16, NTFS, Linux ext2, Linux ext2, Solaris UFS, etc).

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# fsstat -f fat16 usbimage-fat16
FILE SYSTEM INFORMATION
-----
File System Type: FAT

OEM Name: MSWIN4.1
Volume ID: 0x0
Volume Label (Boot Sector): NO NAME
Volume Label (Root Directory):
File System Type Label: FAT16

Sectors before file system: 32

File System Layout (in sectors)
Total Range: 0 - 121918
* Reserved: 0 - 0
** Boot Sector: 0
* FAT 0: 1 - 239
* FAT 1: 240 - 478
* Data Area: 479 - 121918
** Root Directory: 479 - 510
** Cluster Area: 511 - 121918

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
[root@LinuxForensics gcfa]#
```

This shows the file system type of this image file.

This shows the size of this image file (121919 sectors).

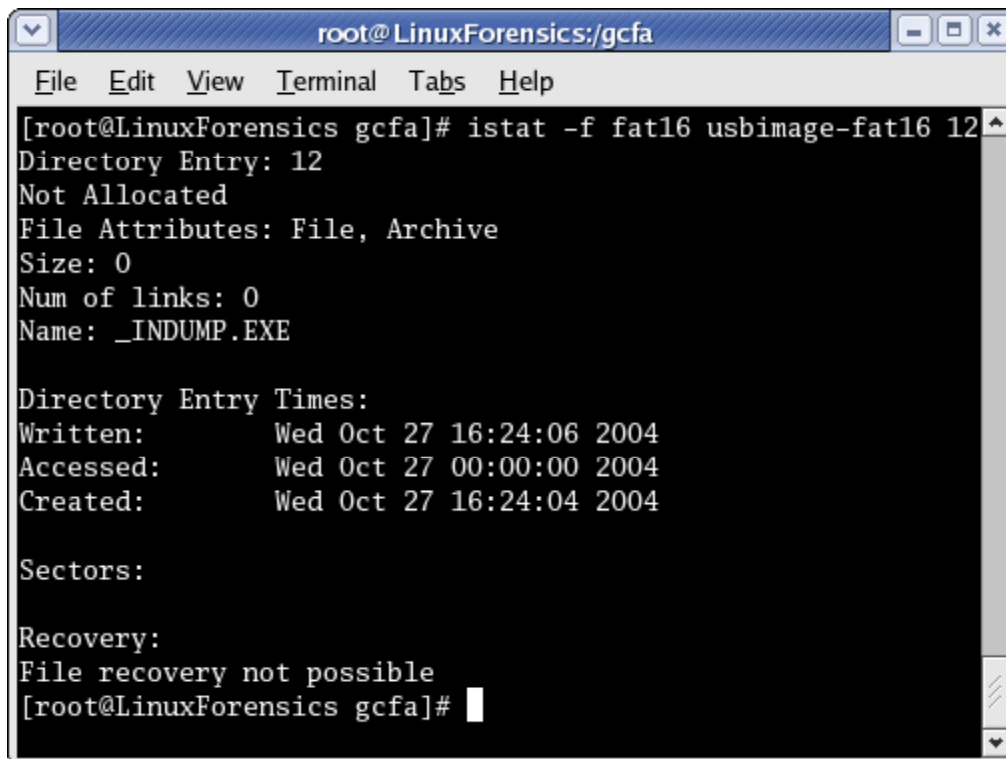
This shows the size of each sector (block) 512 bytes.

This section shows the content on this image. There were three files on this image.

Figure 15 – Fsstat of the FAT16 image

The **istat** utility was used to extract the inode information based on the inode address. There were two inode addresses (Table 1) associated with the WinDump.exe file. The first inode address **12** was tested. The **istat** utility was run with the following options:

- **-f** – this option specifies the image file system type (eg. FAT16, NTFS, Linux ext2, Linux ext2, Solaris UFS, etc).



```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# istat -f fat16 usbimage-fat16 12
Directory Entry: 12
Not Allocated
File Attributes: File, Archive
Size: 0
Num of links: 0
Name: _INDUMP.EXE

Directory Entry Times:
Written:      Wed Oct 27 16:24:06 2004
Accessed:     Wed Oct 27 00:00:00 2004
Created:      Wed Oct 27 16:24:04 2004

Sectors:

Recovery:
File recovery not possible
[root@LinuxForensics gcfa]#
```

Figure 16 – Istat on inode address 12 of the FAT16 image

Figure 16 shows that the file recovery of the inode address **12** was not possible. The second inode address **14** was tested next.

This shows the inode address **14**.

This shows the inode address was not allocated. Therefore, the file associated with this inode address was deleted.

This shows the file size (in bytes).

This shows the name of the file that was associated with this inode address.

This section shows the blocks that this file had occupied.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# istat -f fat16 usbimage-fat16 14
Directory Entry: 14
Not Allocated
File Attributes: File, Archive
Size: 450560
Num of links: 0
Name: _INDUMP.EXE

Directory Entry Times:
Written:      Wed Oct 27 16:24:02 2004
Accessed:     Thu Oct 28 00:00:00 2004
Created:      Wed Oct 27 16:24:04 2004

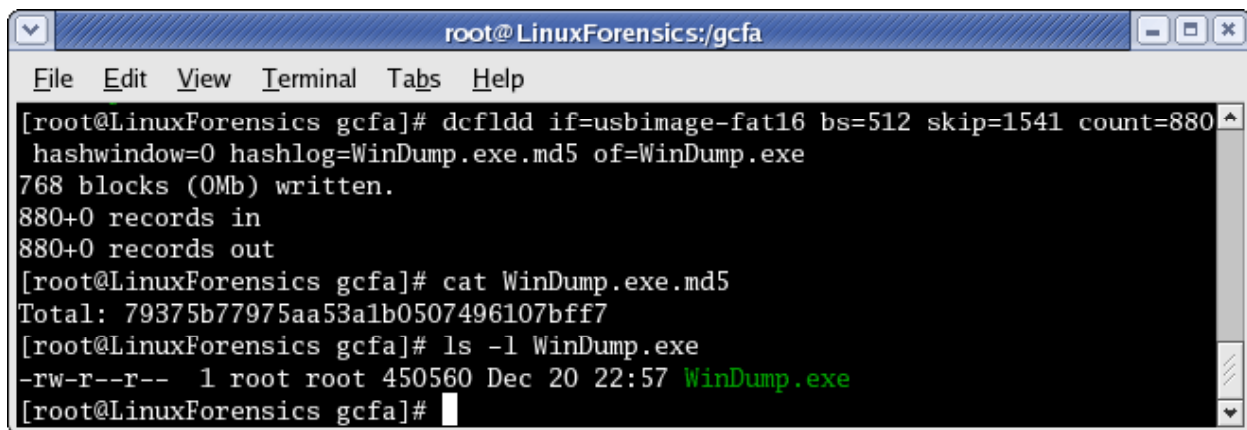
Sectors:
1541 1542

Recovery:
1541 1542 1543 1544 1545 1546 1547 1548
1549 1550 1551 1552 1553 1554 1555 1556
1557 1558 1559 1560 1561 1562 1563 1564
1565 1566 1567 1568 1569 1570 1571 1572
1573 1574 1575 1576 1577 1578 1579 1580
1581 1582 1583 1584 1585 1586 1587 1588
1589 1590 1591 1592 1593 1594 1595 1596
1597 1598 1599 1600 1601 1602 1603 1604
1605 1606 1607 1608 1609 1610 1611 1612
1613 1614 1615 1616 1617 1618 1619 1620
1621 1622 1623 1624 1625 1626 1627 1628
1629 1630 1631 1632 1633 1634 1635 1636
1637 1638 1639 1640 1641 1642 1643 1644
```

Figure 17 – Istat on inode address 12 of the FAT16 image

Figure 17 shows that file recovery for inode address **14** was possible. Figure 17 also shows the starting and ending blocks of this file (**1541** and **2420**). Thus, the “windump.exe” file contained **880** blocks (the number of blocks from 1521 to 2420). The size of the file was **450560** bytes. With this information, the **dcfldd** utility was used to extract this file from the image.

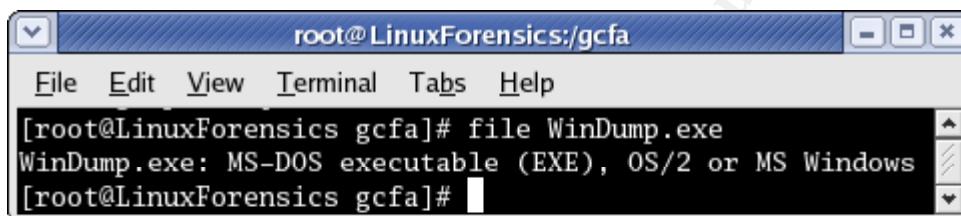




```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# dcfldd if=usbimage-fat16 bs=512 skip=1541 count=880
hashwindow=0 hashlog=WinDump.exe.md5 of=WinDump.exe
768 blocks (0Mb) written.
880+0 records in
880+0 records out
[root@LinuxForensics gcfa]# cat WinDump.exe.md5
Total: 79375b77975aa53a1b0507496107bff7
[root@LinuxForensics gcfa]# ls -l WinDump.exe
-rw-r--r-- 1 root root 450560 Dec 20 22:57 WinDump.exe
[root@LinuxForensics gcfa]#
```

Figure 18 – Dcfldd of the WinDump.exe file

Figure 18 shows the first recovered “WinDump.exe” file. The size of the recovered file matched the file size indicated from the timeline and the istat utility in Figure 17. The MD5 hash of this recovered file will be used later in the investigation to locate this file on the Internet.



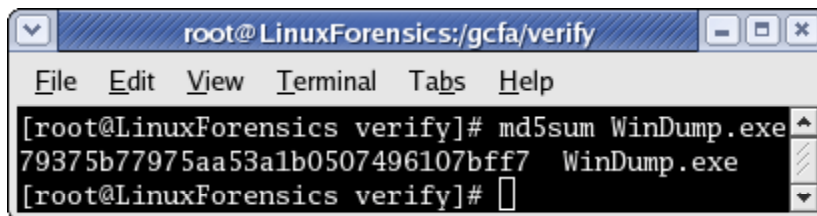
```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# file WinDump.exe
WinDump.exe: MS-DOS executable (EXE), OS/2 or MS Windows
[root@LinuxForensics gcfa]#
```

Figure 19 – File type classification of the WinDump.exe file

The file extension indicated that the “WinDump.exe” was an executable file. To verify the file classification, the **file** utility was used. Figure 19 confirmed that the recovered “WinDump.exe” file was a DOS or Windows executable file.

At this point of the examination, the assumption was that the “WinDump.exe” might be a Windows version of the **tcpdump** utility. The **tcpdump** utility is a Linux sniffer. It is used to capture (intercept) network packets on a network segment. The next step was to use the Google search engine to find a copy of “WinDump.exe”. “WinDump.exe” was used as the search parameter in the Google search engine. The search led to a site <http://windump.polito.it/install/default.htm>. This site contains the WinDump.exe version 3.8.3 beta executable file and the source code (as of January 3, 2005). Both of these files were downloaded and copied to the /gcfa/verify folder of the Linux forensic machine.

The **md5sum** utility was used next to calculate the MD5 hash of the downloaded “WinDump.exe” file. The MD5 hash of this file would then be compared to the MD5 hash of the recovered “WinDump.exe” from the unknown USB JumpDrive image. The result would indicate if both files were the same file.

A terminal window titled 'root@LinuxForensics:/gcfa/verify'. The command prompt shows '[root@LinuxForensics verify]# md5sum WinDump.exe'. The output is '79375b77975aa53a1b0507496107bff7 WinDump.exe'. The prompt then shows '[root@LinuxForensics verify]# ' with a cursor.

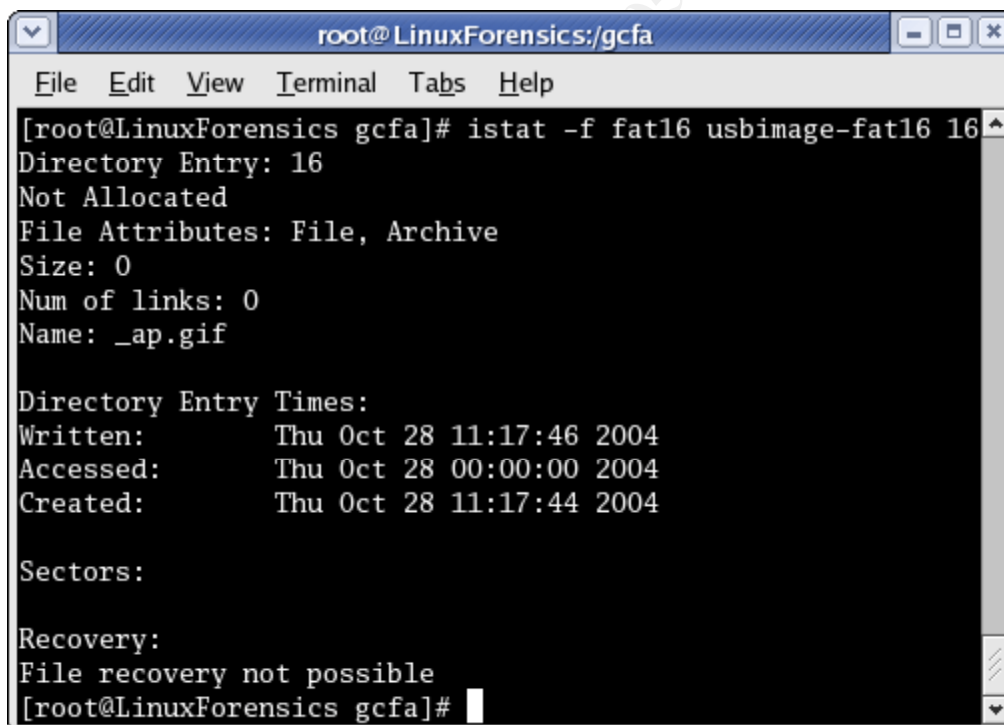
```
root@LinuxForensics:/gcfa/verify
File Edit View Terminal Tabs Help
[root@LinuxForensics verify]# md5sum WinDump.exe
79375b77975aa53a1b0507496107bff7 WinDump.exe
[root@LinuxForensics verify]#
```

Figure 20 – MD5 hash of the downloaded WinDump.exe file

The MD5 hash of the downloaded “WinDump.exe” file (Figure 20) matched the MD5 hash of the recovered “WinDump.exe” file (Figure 18). It was concluded that the recovered “WinDump.exe” file was a Windows sniffer. At this point of the investigation, it was assumed that Mr. Lawrence had used the WinDump program to capture network packets. However, it was still unknown as to why he chose to initiate this action.

### ap.gif

The next step in the investigation was to recover the “\_ap.gif” file. The same recovery procedures were used. There were two inode addresses that were associated to this file (Table 1). The inode address **16** was tested first.

A terminal window titled 'root@LinuxForensics:/gcfa'. The command prompt shows '[root@LinuxForensics gcfa]# istat -f fat16 usbimage-fat16 16'. The output shows details for Directory Entry 16, including file attributes, size, links, name, and times. It concludes with 'Recovery: File recovery not possible'. The prompt then shows '[root@LinuxForensics gcfa]# ' with a cursor.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# istat -f fat16 usbimage-fat16 16
Directory Entry: 16
Not Allocated
File Attributes: File, Archive
Size: 0
Num of links: 0
Name: _ap.gif

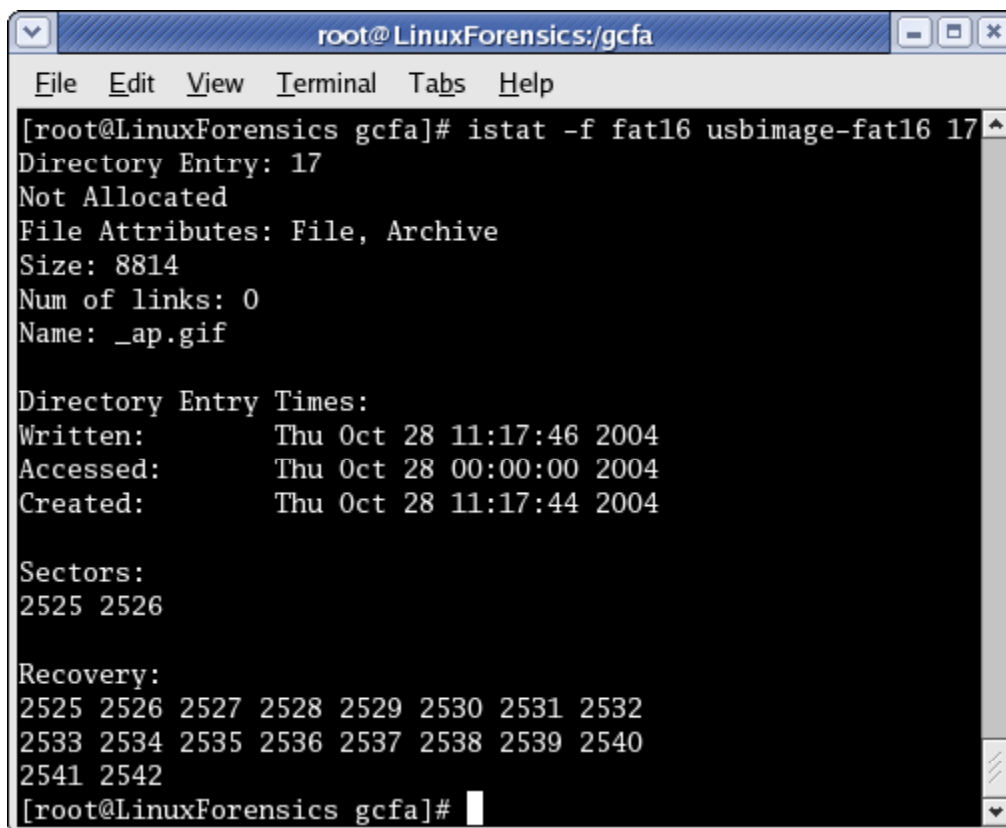
Directory Entry Times:
Written:      Thu Oct 28 11:17:46 2004
Accessed:    Thu Oct 28 00:00:00 2004
Created:     Thu Oct 28 11:17:44 2004

Sectors:

Recovery:
File recovery not possible
[root@LinuxForensics gcfa]#
```

Figure 21 – Istat on inode address 16 of the FAT16 image

Figure 21 shows that the file recovery of the inode address **16** was not possible. The second inode address **17** was tested next.



```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# istat -f fat16 usbimage-fat16 17
Directory Entry: 17
Not Allocated
File Attributes: File, Archive
Size: 8814
Num of links: 0
Name: _ap.gif

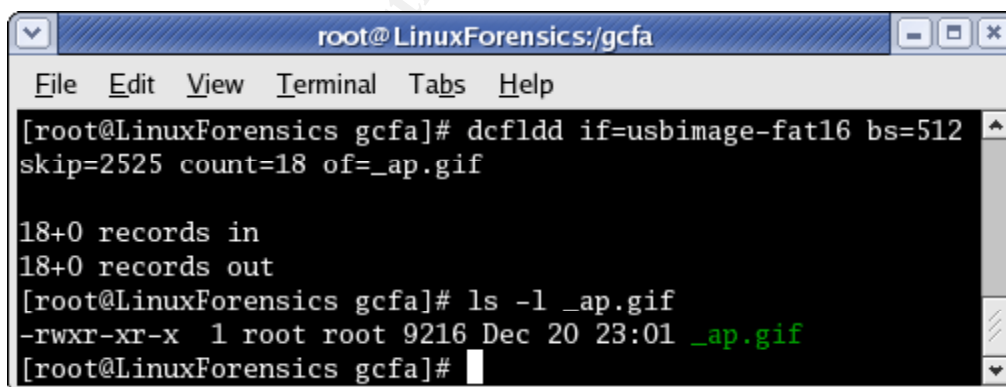
Directory Entry Times:
Written: Thu Oct 28 11:17:46 2004
Accessed: Thu Oct 28 00:00:00 2004
Created: Thu Oct 28 11:17:44 2004

Sectors:
2525 2526

Recovery:
2525 2526 2527 2528 2529 2530 2531 2532
2533 2534 2535 2536 2537 2538 2539 2540
2541 2542
[root@LinuxForensics gcfa]#
```

Figure 22 – Istat on inode address 17 of the FAT16 image

Figure 22 shows that the file recovery for the inode address **17** was possible. This figure also showed the starting and ending blocks of this file (**2525** and **2542**). Thus, this file contained **18** blocks (the number of blocks from 2525 to 2542). The size of the file was **8814** bytes. With this information, the **dcfldd** utility was used to extract this file from the image.



```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# dcfldd if=usbimage-fat16 bs=512
skip=2525 count=18 of=_ap.gif

18+0 records in
18+0 records out
[root@LinuxForensics gcfa]# ls -l _ap.gif
-rwxr-xr-x 1 root root 9216 Dec 20 23:01 _ap.gif
[root@LinuxForensics gcfa]#
```

Figure 23 – Dcfldd of the \_ap.gif file

Figure 23 shows that the recovered file size did not match the file size indicated from the **istat** utility (Figure 22) and the timeline (Figure 14). This was due to the fact that the actual file “\_ap.gif” required only 17.21484375 blocks (the actual size of the “\_ap.gif”

8814 bytes divided by the block size 512 bytes). However, it required a total of 18 blocks to contain this file because each block is 512 bytes. Therefore, 18 blocks were extracted by the **dcflddd** utility (18 blocks multiple by 512 bytes = the size of the “\_ap.gif” file 9216 bytes). Thus, the actual blocks required by the file were subtracted from the number of blocks extracted to give the blocks that were not used by the \_ap.gif file (18 - 17.21484375 = 0.78515625). This result was multiplied by the block size (512 bytes) to obtain the unused space in bytes (0.78515625 \* 512 = 402 bytes).

The **khxedit** utility was used to remove the unused space in the “\_ap.gif” file.

```
[root@LinuxForensics gcfa]# khxedit _ap.gif
```

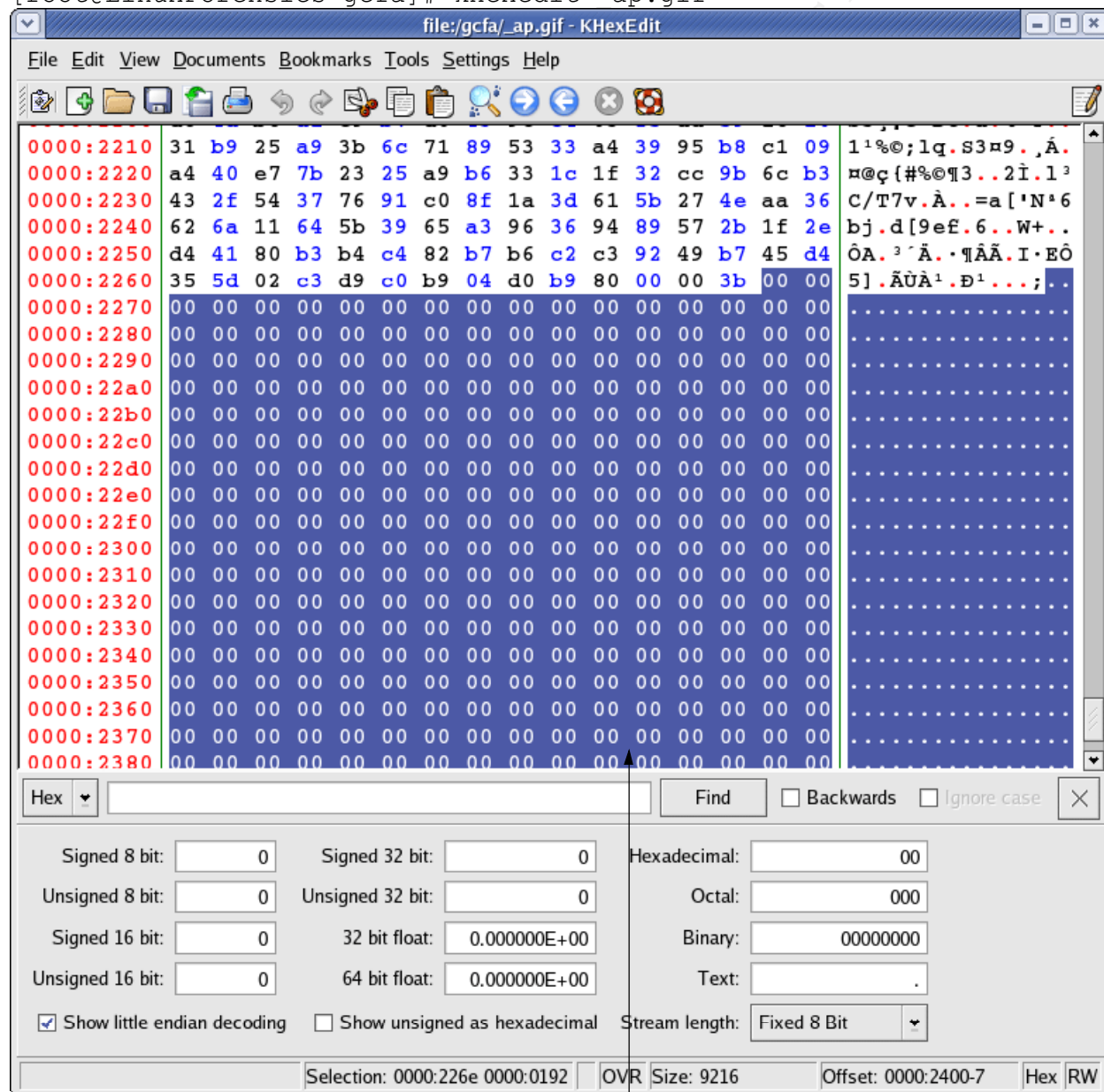


Figure 24 – Khexdit of the recovered \_ap.gif file

The highlighted area that contained “00” was deleted.

The unused space was located at the end of the file. Each hexadecimal digit represents 4 bits of data. Therefore, 2 hexadecimal digits represent 1 byte of data since 8 bits equals 1 byte. Therefore, the total number of hexadecimal digits deleted from the “\_ap.gif” file was 804. The file was saved after the modification. Figure 25 shows that the modified \_ap.gif file was 8814 bytes.

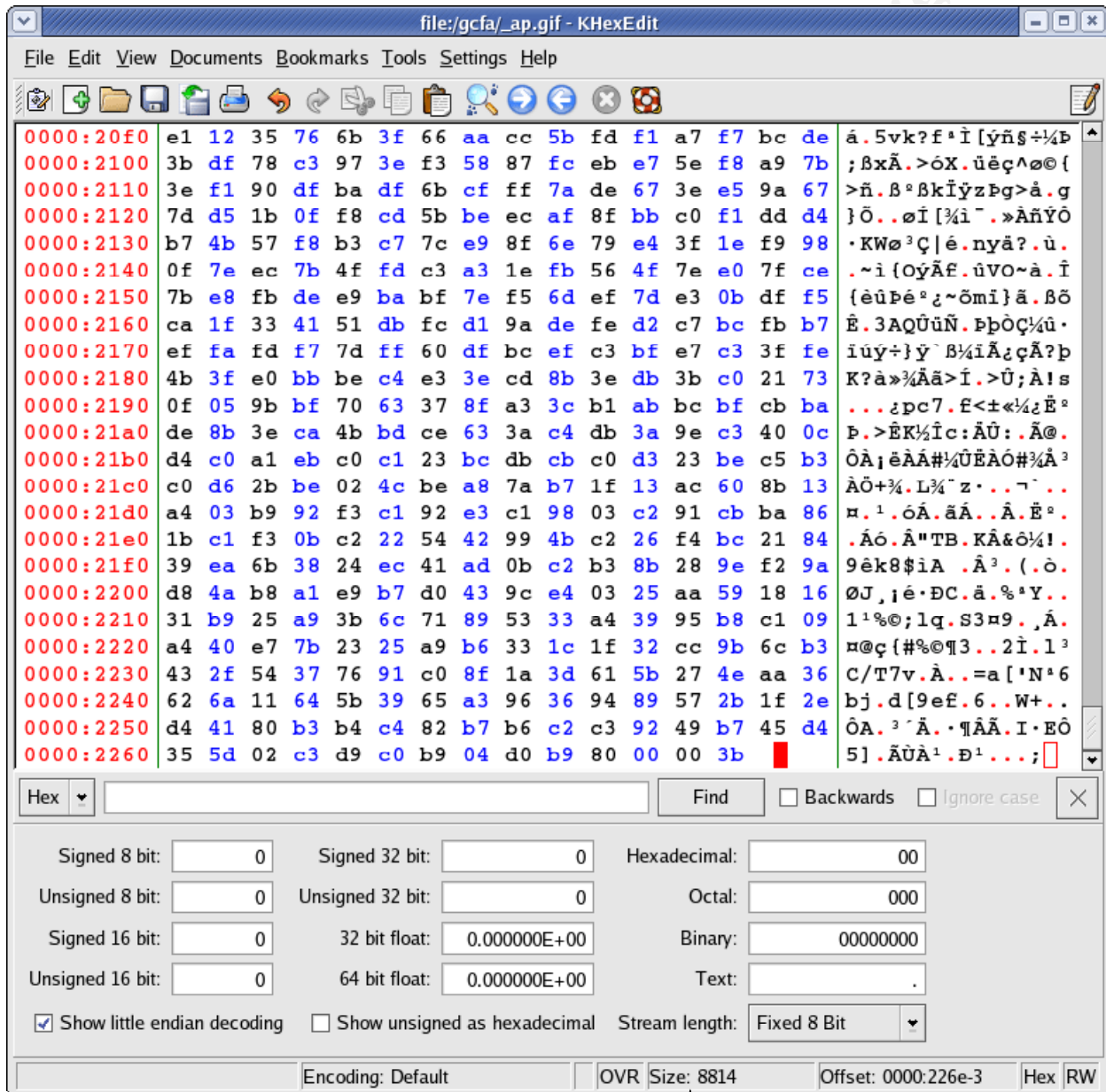
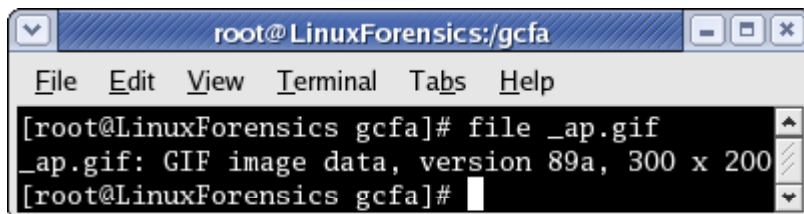


Figure 25 – Khexdit of the modified \_ap.gif file



The file extension of the “\_ap.gif” indicated that it was a GIF file. The **file** utility was used to verify this.

A terminal window titled 'root@LinuxForensics:/gcfa' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command prompt shows '[root@LinuxForensics gcfa]# file \_ap.gif' followed by the output '\_ap.gif: GIF image data, version 89a, 300 x 200'. The prompt then returns to '[root@LinuxForensics gcfa]#'.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# file _ap.gif
_ap.gif: GIF image data, version 89a, 300 x 200
[root@LinuxForensics gcfa]#
```

Figure 26 – File type classification on the \_ap.gif file

The **file** utility in Figure 26 confirmed that the “\_ap.gif” file was a GIF file. The **xview** utility was used next to view the “\_ap.gif”.

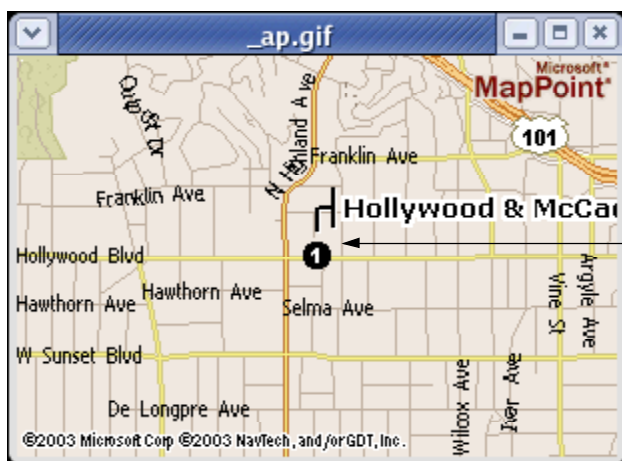
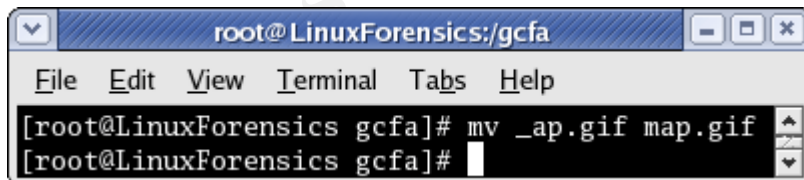


Figure 27 – The Map

Figure 27 shows that the “\_ap.gif” file was a map. It was assumed that Mr. Lawrence had used <http://www.mapblast.com> to locate the address “Hollywood & McCa”. However, the cross street “McCa” was unknown at this point. It was concluded that the actual name of the “\_ap.gif” was “map.gif” thus the file was renamed to “map.gif”.

A terminal window titled 'root@LinuxForensics:/gcfa' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command prompt shows '[root@LinuxForensics gcfa]# mv \_ap.gif map.gif' followed by the prompt '[root@LinuxForensics gcfa]#'.

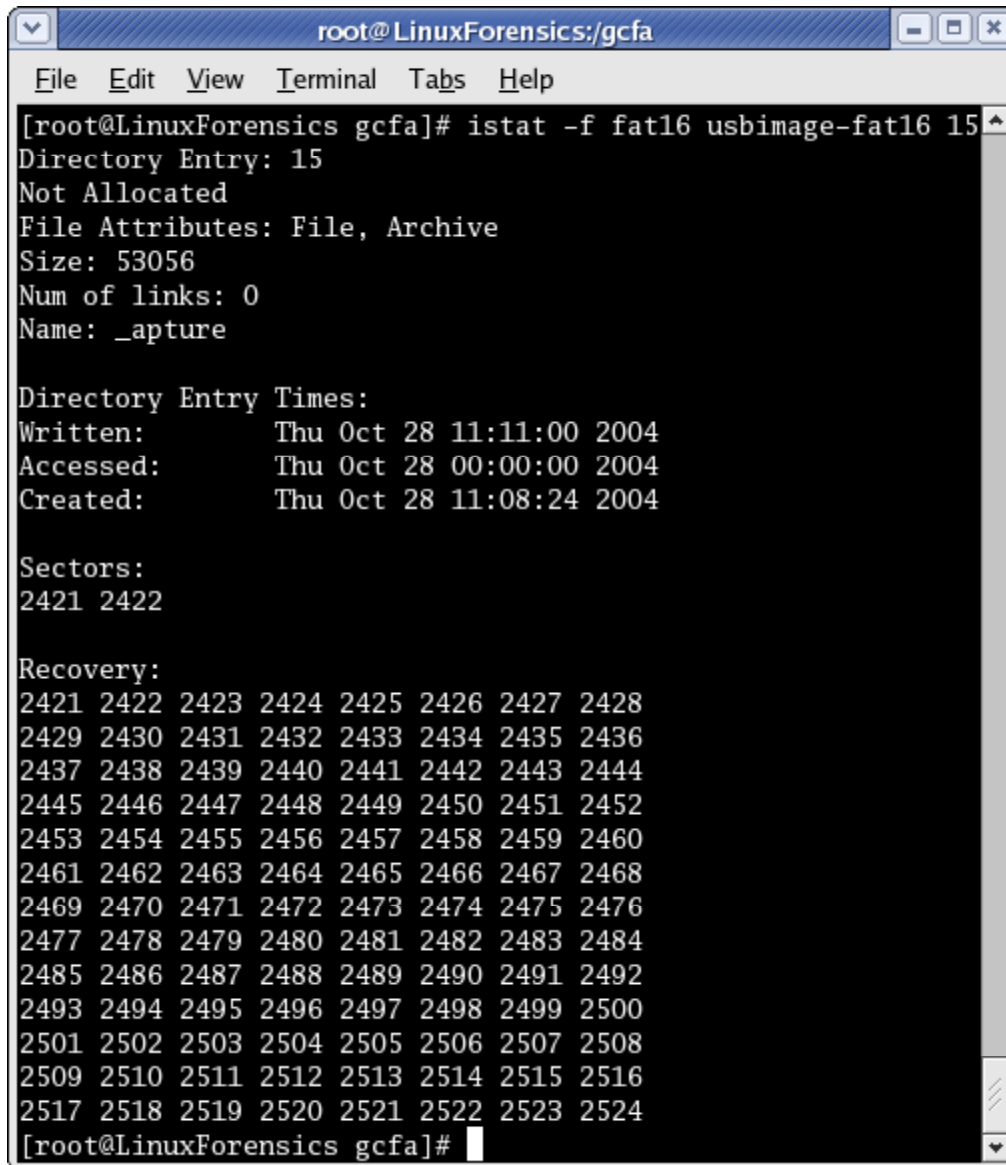
```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# mv _ap.gif map.gif
[root@LinuxForensics gcfa]#
```

Figure 28 – Rename of the \_ap.gif file to map.gif

It was concluded at this point of the investigation that Mr. Lawrence had some interest in this location. However, the reason for his interest was still unknown.

## apture

The next step in the investigation was to recover the “\_apture” file. There was only one inode address **15** associated (Table 1) to this file therefore the **istat** utility was run on this inode address.



```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# istat -f fat16 usbimage-fat16 15
Directory Entry: 15
Not Allocated
File Attributes: File, Archive
Size: 53056
Num of links: 0
Name: _apture

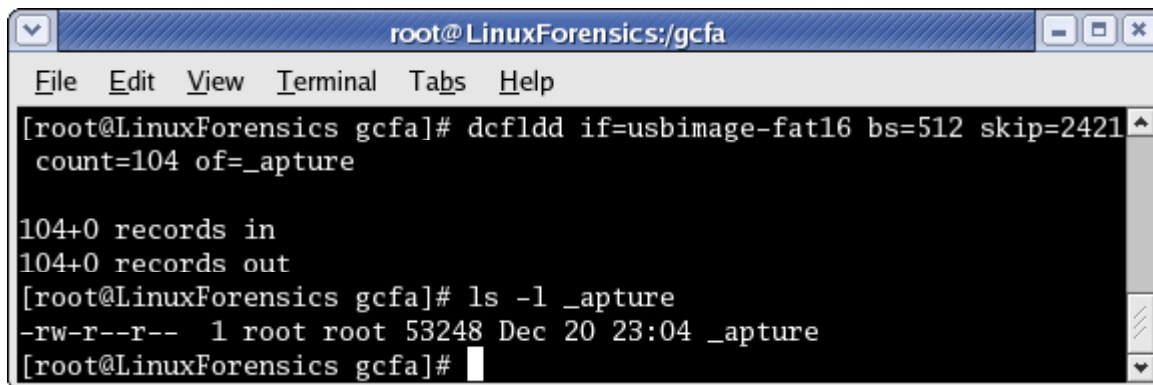
Directory Entry Times:
Written: Thu Oct 28 11:11:00 2004
Accessed: Thu Oct 28 00:00:00 2004
Created: Thu Oct 28 11:08:24 2004

Sectors:
2421 2422

Recovery:
2421 2422 2423 2424 2425 2426 2427 2428
2429 2430 2431 2432 2433 2434 2435 2436
2437 2438 2439 2440 2441 2442 2443 2444
2445 2446 2447 2448 2449 2450 2451 2452
2453 2454 2455 2456 2457 2458 2459 2460
2461 2462 2463 2464 2465 2466 2467 2468
2469 2470 2471 2472 2473 2474 2475 2476
2477 2478 2479 2480 2481 2482 2483 2484
2485 2486 2487 2488 2489 2490 2491 2492
2493 2494 2495 2496 2497 2498 2499 2500
2501 2502 2503 2504 2505 2506 2507 2508
2509 2510 2511 2512 2513 2514 2515 2516
2517 2518 2519 2520 2521 2522 2523 2524
[root@LinuxForensics gcfa]#
```

Figure 29 – Istat on inode address 15 of the FAT16 image

Figure 29 shows that file recovery for inode address **15** was possible. Figure 29 also shows the starting and ending blocks of this file (**2421** and **2524**). Thus, this file contained **104** blocks (the number of blocks from 2421 to 2524). The size of the file was **53056** bytes. With this information, the **dcfldd** utility was used to extract this file from the image.



```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# dcfldd if=usbimage-fat16 bs=512 skip=2421
count=104 of=_apture

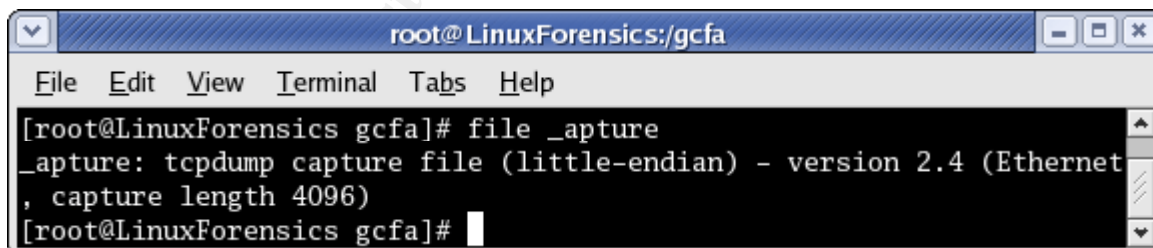
104+0 records in
104+0 records out
[root@LinuxForensics gcfa]# ls -l _apture
-rw-r--r-- 1 root root 53248 Dec 20 23:04 _apture
[root@LinuxForensics gcfa]#
```

Figure 30 – Dcfldd of the \_apture file

Figure 30 shows the recovered file size did not match the file size indicated by the **istat** utility and the timeline. The same procedures that were used to determine and delete the unused space for the file “map.gif” were used again. The actual file “\_apture” requires only 103.625 blocks (the actual size of the \_apture 53056 bytes divided by the block size 512 bytes). However, it required a total of 104 blocks to contain this file because each block is 512 bytes. Therefore, 104 blocks were extracted by the **dcfldd** utility. The actual blocks required by the file were subtracted from the number of blocks extracted to give the blocks that were not used by the “\_apture” file ( $104 - 103.625 = 0.375$ ). This result was then multiplied by the block size (512 bytes) to obtain the unused space in bytes ( $0.375 * 512 = 192$  bytes).

The **khxedit** utility was used next to remove the unused space. The total number of hexadecimal digits (192 bytes multiple by 2 = 384) was deleted from the “\_apture” file. The file was saved after the modification. The size of the “\_apture” file after the modification was 53056 bytes.

The **file** utility was used to determine the file type of the “\_apture” recovered file.



```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# file _apture
_apture: tcpdump capture file (little-endian) - version 2.4 (Ethernet
, capture length 4096)
[root@LinuxForensics gcfa]#
```

Figure 31 – File type classification of the \_apture file

Figure 31 shows that the “\_apture” file was a tcpdump capture file. The file was renamed to “capture” as showed in Figure 32.



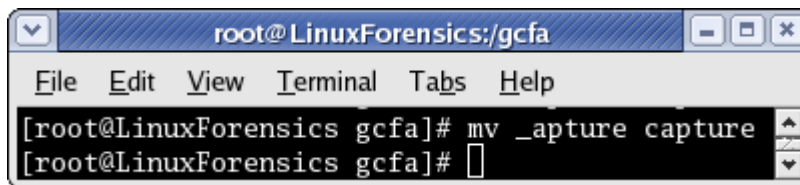


Figure 32 – Rename of the \_apture file to capture

The **ethereal** utility was used to further examine this “capture” file. **Ethereal** is a GUI based network sniffer, which can be used to capture network packets on a network segment. It also contains options to allow a particular TCP data stream to be followed, filtering by destination or source addresses, filtering by particular TCP or UDP port numbers, displaying the captured data and many capabilities to analyze captured network packets. The **ethereal** utility can also read tcpdump network captures. Figure 33 shows the capture file in the **ethereal** utility.

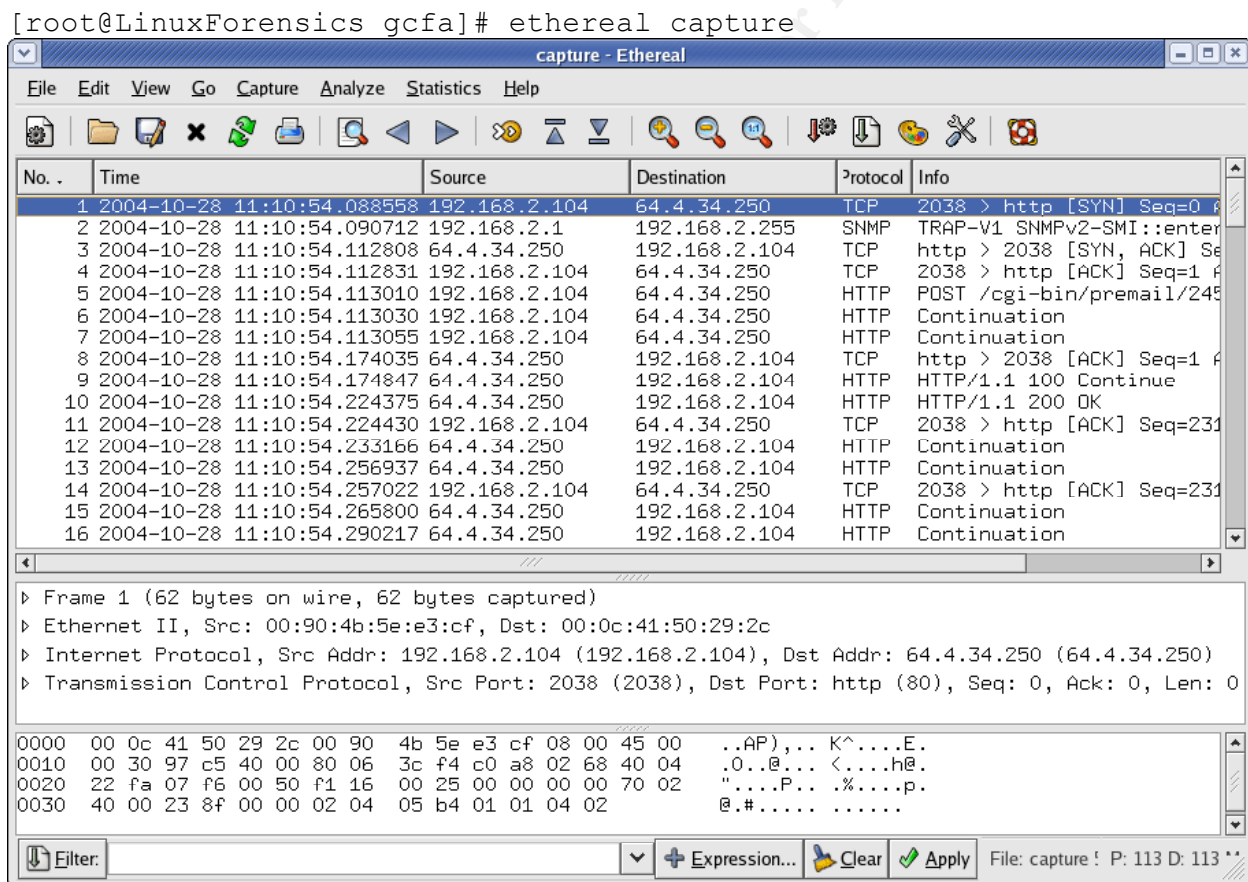


Figure 33 – Ethereal of the capture file

The first step in the analysis of the capture file was to change the time format. The initial time format was based on the seconds since the beginning of the capture. The time format was changed to date and time of the day. Immediately, it was concluded that that this network capture started on October 28<sup>th</sup> 2004 at 11:10:54 am.

The next step was to investigate each TCP data streams to determine what information was captured. A TCP data stream consists of the following:

- A TCP session 3-way handshake to establish the TCP session between the client and the server. This 3-way handshake is as follow:
  - The client sends a SYN packet to the server.
  - The server responds with a SYN, ACK packet to the client.
  - The client responds with an ACK packet to the server.
- Data that is passed in this session.
- A TCP session completion to indicate that the TCP session between the client and the server is finished. This TCP session completion is as follow:
  - The client sends a FIN, ACK packet to the server.
  - The server responds with an ACK packet to the client.

The **ethereal** utility can follow the TCP data stream by first looking for the TCP session 3-way handshake. The TCP session 3-way handshake indicates the beginning of a TCP data stream. Once the 3-way handshake has completed, the **ethereal** utility looks at the TCP Protocol data. The TCP Protocol data contains information such as the source port and destination port, the actual data payload, and the Sequence Number, Acknowledge Number and the Next Sequence Number. Therefore, each TCP packet (other than the initial handshake and session completion) has a sequence number that represents this TCP packet, an Acknowledge number that acknowledges the previous TCP packet, and a Next Sequence Number that indicates the next TCP packet. With this information, the **ethereal** utility can follow a particular TCP stream and display the data inside this TCP stream.

The first packet was highlighted using the right-click mouse button and the “Follow TCP Stream” option was selected.



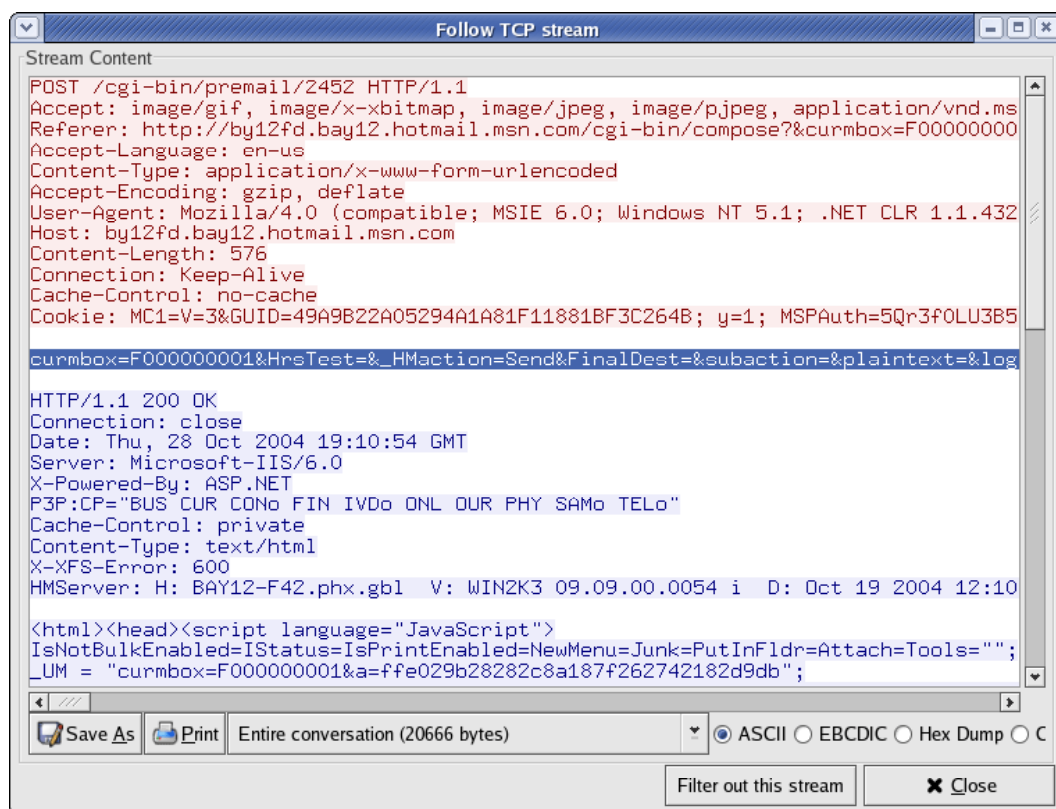


Figure 34 – The TCP Data Stream

The examination of the TCP stream (Figure 34) concluded with the following:

- This was a network capture of a Hotmail session. Hotmail is a Microsoft product that provides web-based email services.
- The highlighted area in Figure 34 showed the activity of the current mailbox “curmbox”. This highlighted area is displayed below:

curmbox=F000000001&HrsTest=&\_Hmaction=Send&FinalDest=&subaction=&plaintext=&login=flowergirl96&msg=&start=&len=&attfile=&attlistfile=&eurl=&type=&src=&ref=&ru=&msgid=b16479b18beec291196189c78555223c\_1098692452&RTBgcolor=&encodedto=SamGuarillo@hotmail.com&encodedbcc=&deleteUponSend=0&importance=&sigflag=&newmail=new&to=SamGuarillo@hotmail.com&cc=&bcc=&subject=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

The current mailbox information showed the following:

- This person was sending an email.
- This person’s hotmail login name was “flowergirl96”. Therefore, her email address was [flowergirl96@hotmail.com](mailto:flowergirl96@hotmail.com)
- The recipient of this message was [SamGuarillo@hotmail.com](mailto:SamGuarillo@hotmail.com)
- The subject of this message was “RE: coffee”

- The body of this message was:  
“Sure, coffee sounds great. Let’s meet at the coffee shop on the corner Hollywood and McCadden. It’s a nice out of the way spot. See you at 7pm.  
–Leila”.
- The body of the message was littered with these %3A, %2C, %27, %0D, %0A. The % indicates Unicode and they translate as follows [3]:
  - 3A is : (colon)
  - 2C is , (comma)
  - 27 is ‘ (??)
  - 0D is an empty space
  - 0A is an empty space

The flowergirl96 and SamGuarillo@hotmail.com were added to the dirty word list. The dirty word list is a list that contains interesting strings that can be used at a latter time to search through an image file using the **strings** utility. The dirty word list usually grows in size as the investigation progresses.

At this point of the investigation, the following can be concluded:

- From the tcpdump capture, Mr. Lawrence was able to determine Ms. Conlay’s email address. This would explain how he was able to send emails to Miss Conlay’s personal email address.
- Mr. Lawrence was also able to determine Ms. Conlay and Mr. Guarillo’s meeting time and location. This would also explain the existence of the map.gif file, which correctly displayed the meeting location (Hollywood and McCadden).

The rest of the capture file was analyzed and there was no interesting information that could be gathered from it.

### **WinPCap 3 1 beta 3.exe**

The next step was to recover the last deleted file. From the timeline (Figure 14), the file name of this deleted file was “WinPCap\_3\_1\_beta\_3.exe” and the inode addresses 7 and 10 (Table 1) were associated with this file. The **istat** utility was used to determine more information on the inode addresses 7 and 10. The results indicated that the data associated with inode addresses 7 and 10 were not recoverable. However, the inode address 10 did provide some valuable information that could be used to extract some portion of this file.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# istat -f fat16 usbimage-fat16 10
Directory Entry: 10
Not Allocated
File Attributes: File, Archive
Size: 485810
Num of links: 0
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
[root@LinuxForensics gcfa]#
```

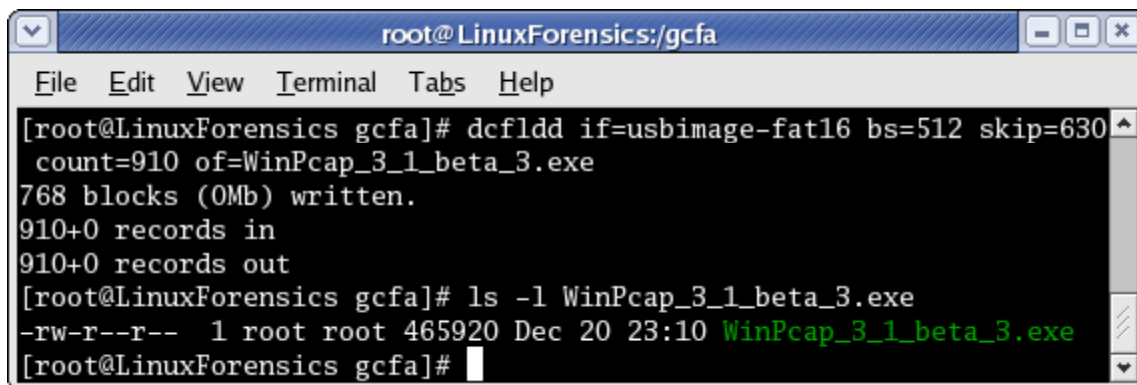
This shows the file size.

This shows the file name.

This shows the file name might have occupied these sectors previously.

Figure 35 – Istat on inode address 10 of the FAT16 image

Figure 35 shows that the file may have occupied the blocks 591 and above. However, the **fsstat** result from Figure 15 shows that blocks 591 to 630 had already been allocated. Therefore, it was concluded a new file had overwritten some parts of the blocks that were occupied by the “WinPcap\_3\_1\_beta\_3.exe” file. Therefore, a partial recovery of this file was attempted. The actual file size was known to be 485810 bytes (the file size was gathered from the **istat** result in Figure 35 and the timeline in Figure 14). Thus, a total of 948.84765625 blocks were needed to contain this file ( $485810/512 = 948.84765625$ ). The file system requires a multiple of 512 bytes blocks to store a file. Therefore, this number was rounded up to 949. The **dcfldd** utility was used to extract this file. The starting block was set to 630 because there was already a file that occupied blocks 591 to 630. This was determined from the **fsstat** results shown in Figure 15. The file that occupied blocks 591 to 630 was “coffee.doc”. This was determined by using the **istat** utility on the “coffee.doc” inode address 18. The “coffee.doc” used a total of 39 blocks (the file size 19968 bytes divided by 512 bytes = 39 blocks). These 39 blocks must be subtracted from the total number of blocks required for the “WinPcap\_3\_1\_beta\_3.exe” file to obtain the remaining size in blocks ( $949 - 39 = 910$ ).



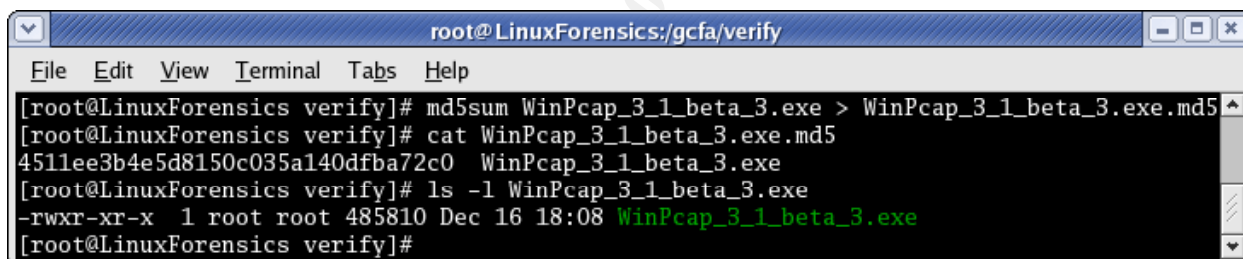
```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# dcfldd if=usbimage-fat16 bs=512 skip=630
count=910 of=WinPcap_3_1_beta_3.exe
768 blocks (0Mb) written.
910+0 records in
910+0 records out
[root@LinuxForensics gcfa]# ls -l WinPcap_3_1_beta_3.exe
-rw-r--r-- 1 root root 465920 Dec 20 23:10 WinPcap_3_1_beta_3.exe
[root@LinuxForensics gcfa]#
```

Figure 36 – Dcfldd of the WinPcap\_3\_1\_beta\_3.exe file

Figure 36 shows the partial file that was extracted by **dcfldd** utility. The next step was to look for a copy of this file on the Internet. The Google search engine was used and the “WinPcap\_3\_1\_beta\_3.exe” string was entered into search field. A copy of the “WinPcap\_3\_1\_beta\_3.exe” was downloaded from this site:

<http://www.oltenia.ro/download/pub/windows/network%20tools/ethereal/>

This downloaded file was copied to the /gcfa/verify folder. The **md5sum** utility was used to calculate the MD5 hash of this file.



```
root@LinuxForensics:/gcfa/verify
File Edit View Terminal Tabs Help
[root@LinuxForensics verify]# md5sum WinPcap_3_1_beta_3.exe > WinPcap_3_1_beta_3.exe.md5
[root@LinuxForensics verify]# cat WinPcap_3_1_beta_3.exe.md5
4511ee3b4e5d8150c035a140dfba72c0 WinPcap_3_1_beta_3.exe
[root@LinuxForensics verify]# ls -l WinPcap_3_1_beta_3.exe
-rwxr-xr-x 1 root root 485810 Dec 16 18:08 WinPcap_3_1_beta_3.exe
[root@LinuxForensics verify]#
```

Figure 37 – MD5 hash of the downloaded WinPcap\_3\_1\_beta\_3.exe file

The size of the downloaded file (Figure 37) was exactly the same as the file size obtained from the **istat** utility (Figure 35) and the timeline (Figure 14). It was assumed at this point that the partially recovered file was the same file as the downloaded file. Since the downloaded file was the complete file, it would contain the missing data that could be used to complete the partially recovered file. Therefore, the plan was to rebuild the partially recovered file by obtaining the missing parts from the downloaded file. Once the missing parts were copied into the partially recovered file, the **md5sum** utility would be used to calculate the MD5 hash of this new recovered file. The MD5 hash would then compared to the MD5 hash of the downloaded file.

The first step was to use the **khxedit** utility to load both the downloaded and the recovered files.

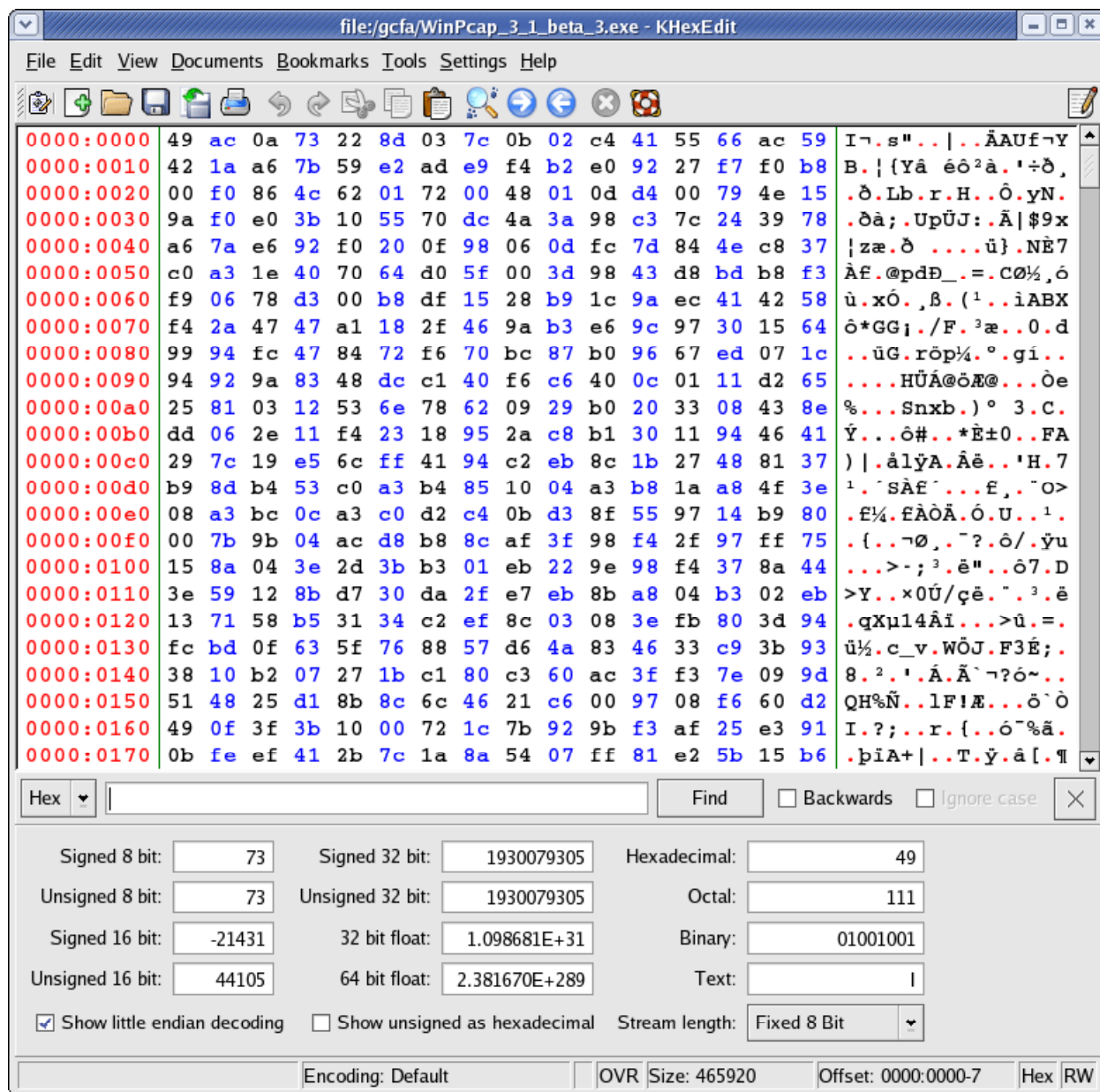


Figure 38 – Khexedit of the partially recovered WinPcap\_3\_1\_beta\_3.exe file

The first 5 bytes of the file shown in Figure 38 were “49 ac 0a 73 22”. These 5 bytes will be used later to determine how much data must be extracted from the downloaded “WinPcap\_3\_1\_beta\_3.exe” file.



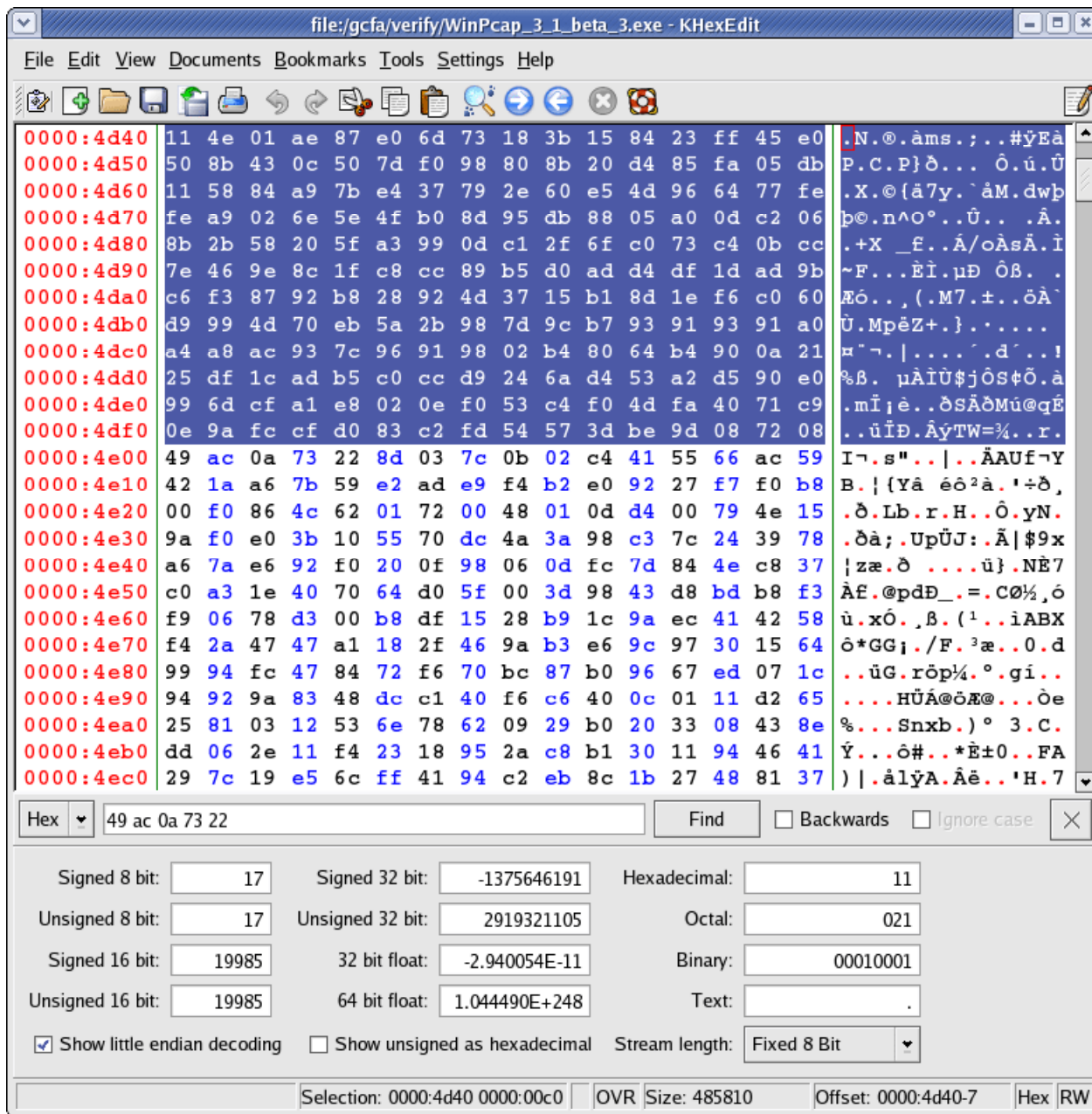


Figure 39 – Khexedit of the downloaded WinPcap\_3\_1\_beta\_3.exe file

The first 5 bytes “49 ac 0a 73 22” of the partially recovered file were entered in the find field in the **Khexedit** of the downloaded WinPcap\_3\_1\_beta. This was done to locate the starting point of the partially recovered file in this downloaded file. Once the first 5 bytes were located, the data prior to these 5 bytes were highlighted (this data was assumed to be the missing data from the partially recovered file). The **Khexedit** copy function in the edit menu was selected. This highlighted data would be pasted into the partially recovered file as shown in Figure 40.





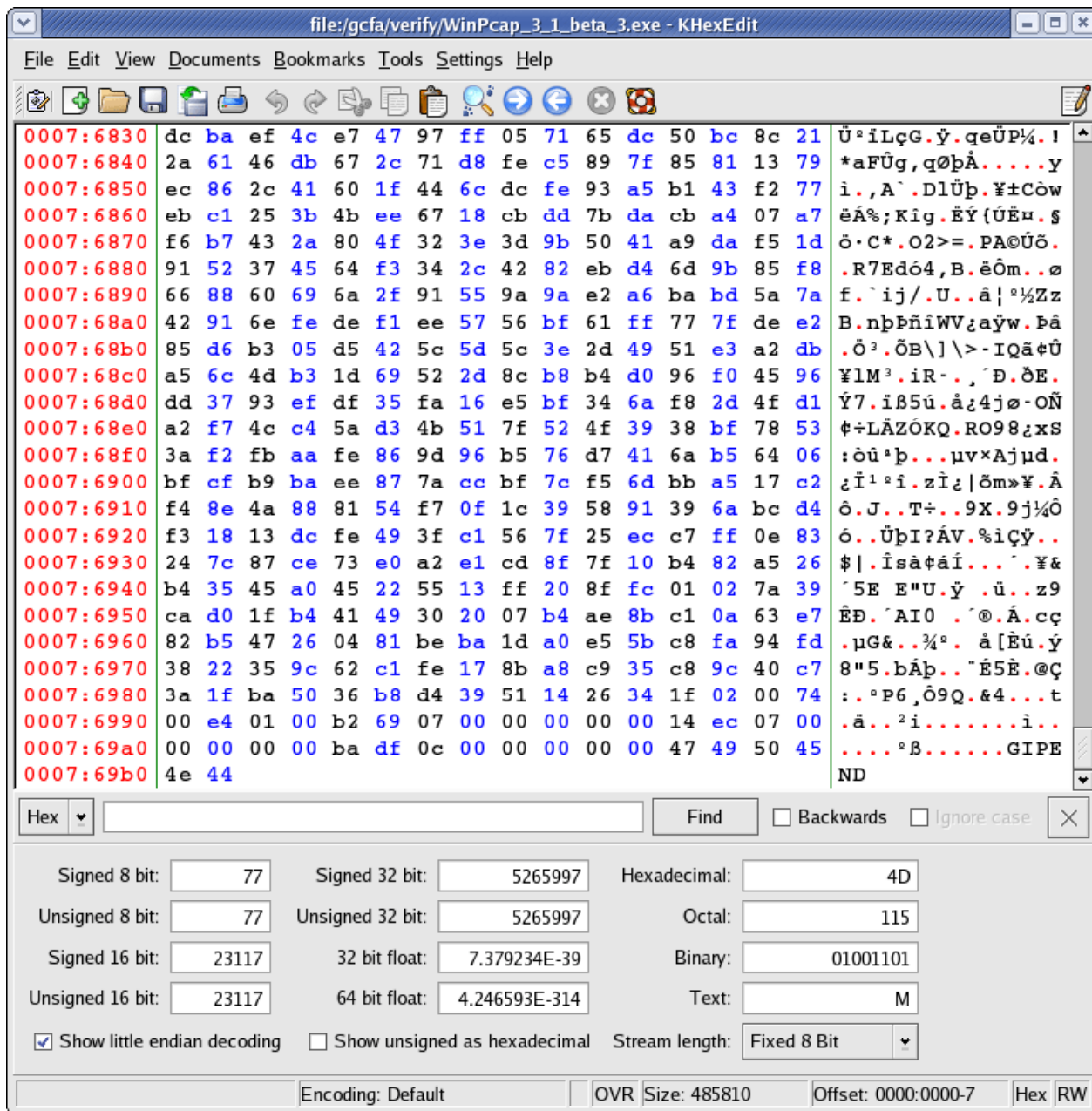


Figure 41 – The end of file (EOF) of the downloaded WinPcap\_3\_1\_beta\_3.exe file

Figure 41 shows the end of the downloaded file. The last 6 bytes of this file are “47 49 50 45 4e 44”. These 6 bytes would be used to locate the EOF of the recovered file.

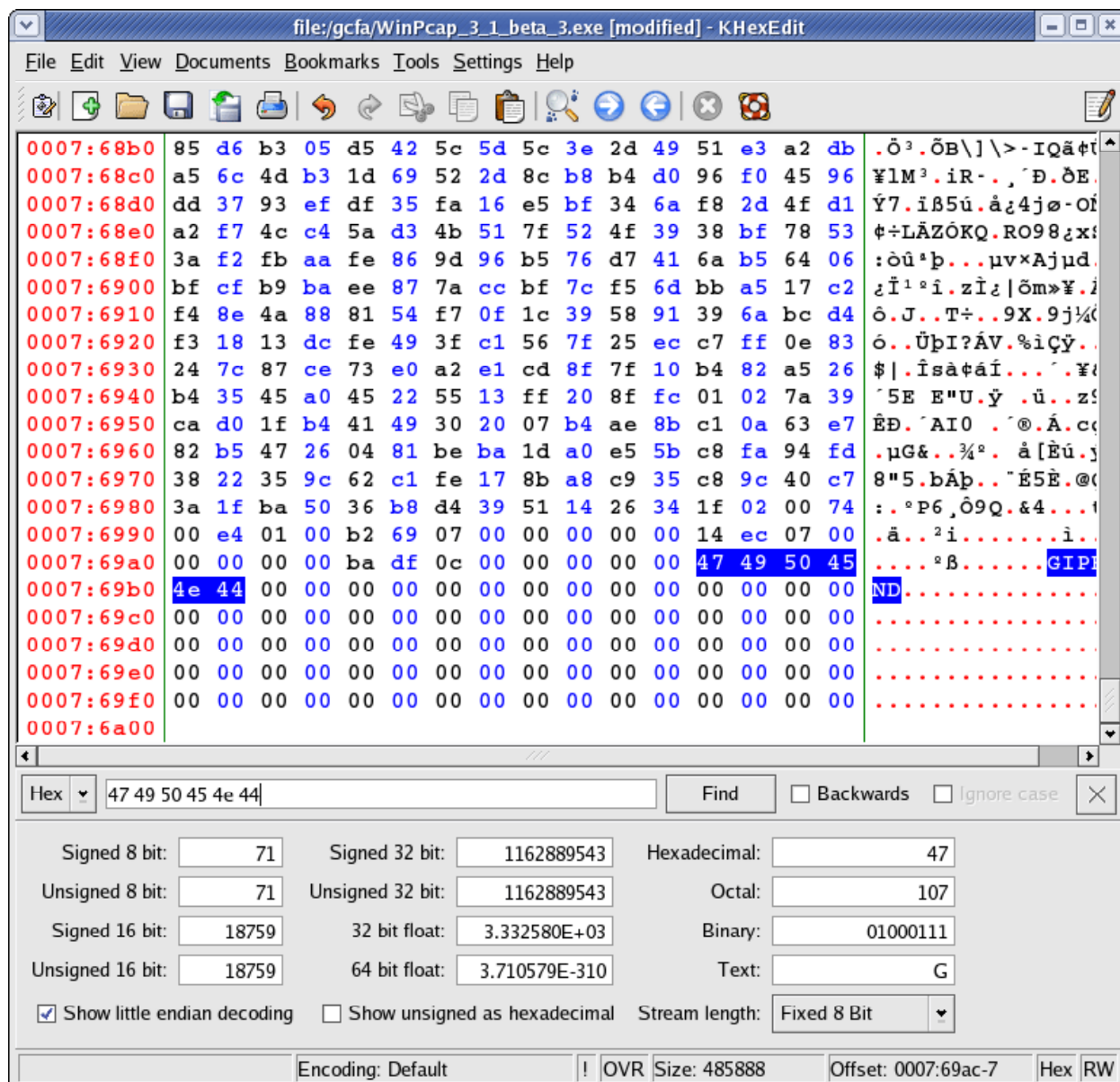


Figure 42 – The end of file (EOF) of the recovered WinPcap\_3\_1\_beta\_3.exe file #1

The last 6 bytes “47 49 50 45 4e 44” of the downloaded file were used to locate the EOF of the recovered file. Figure 42 showed the EOF of the recovered file. The data after the “47 49 50 45 4e 44” were 00s.

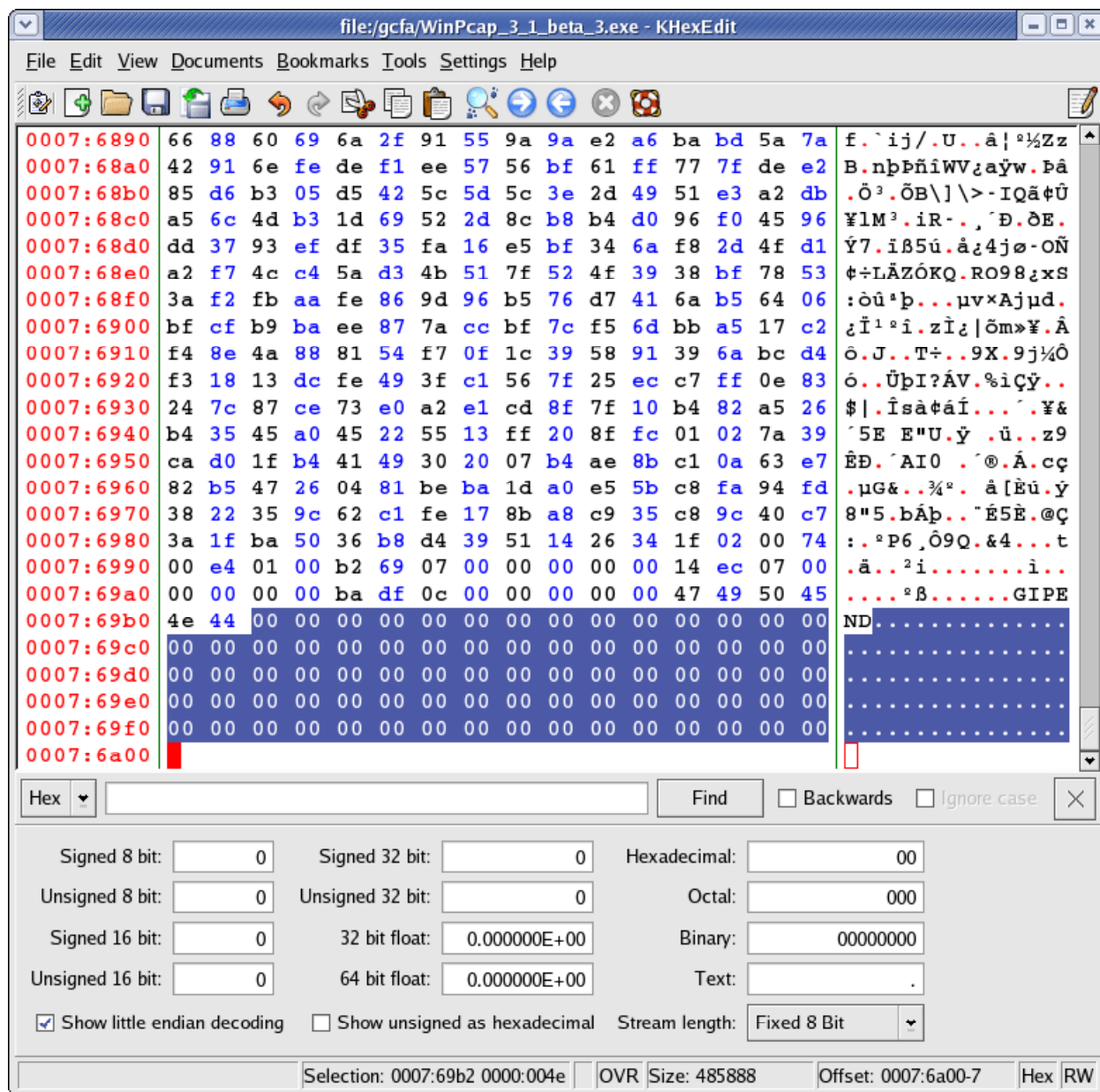
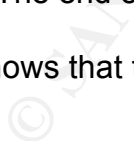


Figure 43 – The end of file (EOF) of the recovered WinPcap\_3\_1\_beta\_3.exe file #2

Figure 43 shows that the unnecessary data (00's) were highlighted and deleted.





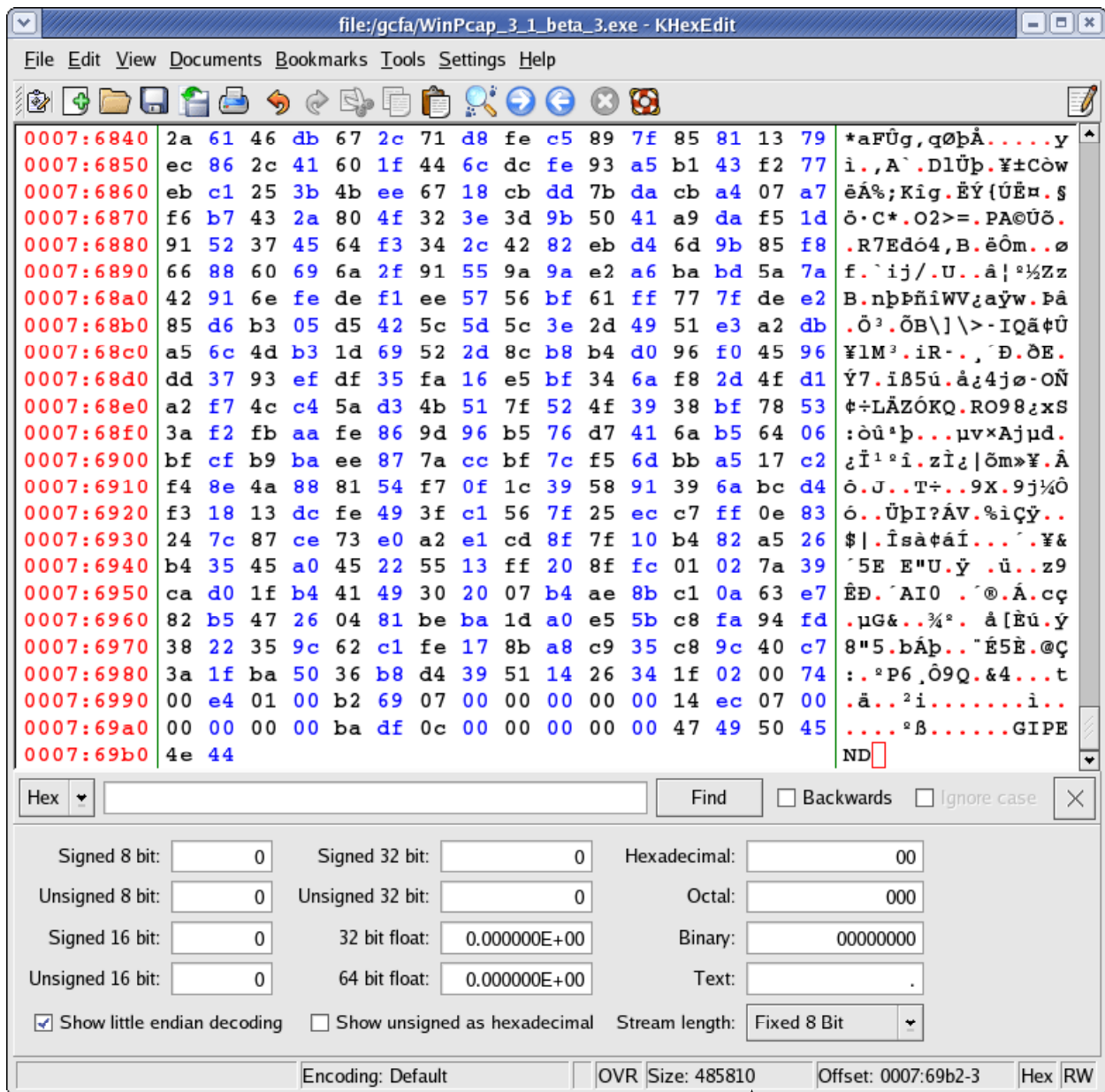
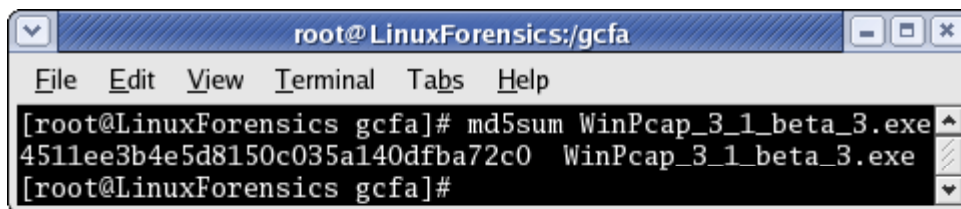


Figure 44 – The complete recovered WinPcap\_3\_1\_beta\_3.exe file

The size of the file showed the correct size (485810 bytes).

The **md5sum** utility was used to calculate the MD5 hash of the complete recovered file. This MD5 hash was then compared to the MD5 hash of the downloaded file to verify if they are the same file.



```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# md5sum WinPcap_3_1_beta_3.exe
4511ee3b4e5d8150c035a140dfba72c0 WinPcap_3_1_beta_3.exe
[root@LinuxForensics gcfa]#
```

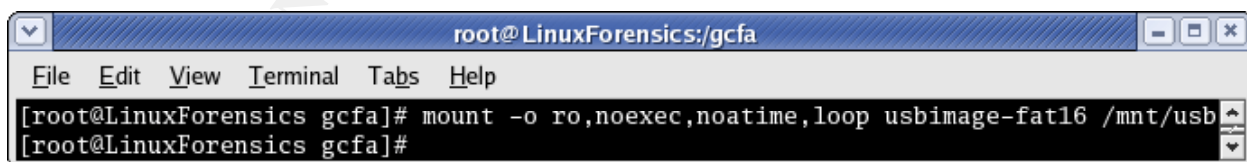
Figure 45 – MD5 hash of the complete recovered WinPcap\_3\_1\_beta\_3.exe file

Figure 45 shows that the MD5 hash of the complete recovered file matched the MD5 hash of the downloaded file. This demonstrated that the partially recovered “WinPcap\_3\_1\_beta\_3.exe” file from the USB JumpDrive image was in fact the downloaded “WinPcap\_3\_1\_beta\_3.exe”.

A further investigation of the purpose of the WinPcap\_3\_1\_beta\_3.exe from <http://windump.polito.it/> revealed that WinPcap contained libraries that are required by WinDump to capture network packets.

Finally, the 3 files that were not deleted were examined. In order to do this, the **mount** utility was used to mount extracted FAT16 partition on /mnt/usb. The **mount** utility was run with the following option:

- **-o ro,noexec,noatime,loop**
  - the **-o** indicates options and these options used were as follow:
    - **ro** - this instructs the **mount** utility to mount the image in read-only mode.
    - **noexec** - this option does not permit executables on the image to be run.
    - **noatime** - this option does not change the accessed time on all the file in the image.
    - **loop** - this option mounts an image on a loop device.



```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# mount -o ro,noexec,noatime,loop usbimage-fat16 /mnt/usb
[root@LinuxForensics gcfa]#
```

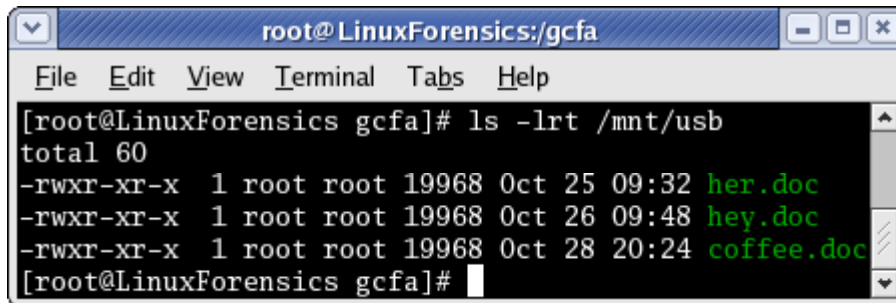
Figure 46 – Mounting the FAT16 image

The **ls** utility was used to list the files on the FAT16 partition. The **ls** utility was run with the options:

- **l** - this option lists the files long format (ie the owners, groups, time, permissions are listed)
- **t** - this option sorts the files based on the last modification file (starting with

the last modified file – descending order)

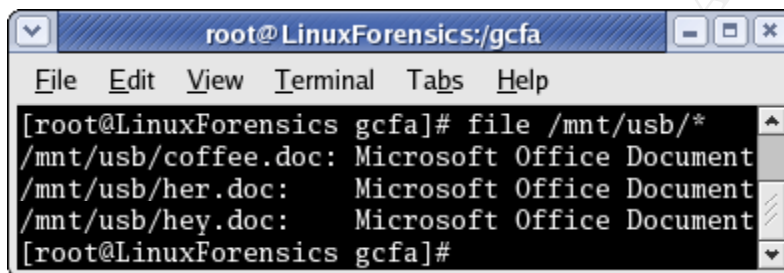
- **r** - this option reverse the sorted order, ie (list the files that were edited the earliest – ascending order).



```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# ls -lrt /mnt/usb
total 60
-rwxr-xr-x 1 root root 19968 Oct 25 09:32 her.doc
-rwxr-xr-x 1 root root 19968 Oct 26 09:48 hey.doc
-rwxr-xr-x 1 root root 19968 Oct 28 20:24 coffee.doc
[root@LinuxForensics gcfa]#
```

Figure 47 – Files on the FAT16 image

The existence of these 3 files validated the timeline gathered from the **fls** utility. The timeline (Figure 14) indicated that 3 files were not deleted. The **ls** listing also indicates that the “her.doc” file was last modified on Oct 25 at 9:32am, the “hey.doc” file was modified on Oct 26 at 9:38am and the “coffee.doc” file was edited at Oct 28 at 20:24.



```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# file /mnt/usb/*
/mnt/usb/coffee.doc: Microsoft Office Document
/mnt/usb/her.doc: Microsoft Office Document
/mnt/usb/hey.doc: Microsoft Office Document
[root@LinuxForensics gcfa]#
```

Figure 48 – File type classification of the files on the FAT16 image

Figure 48 shows that these files were Microsoft Office documents. These documents were therefore loaded into Microsoft Word.

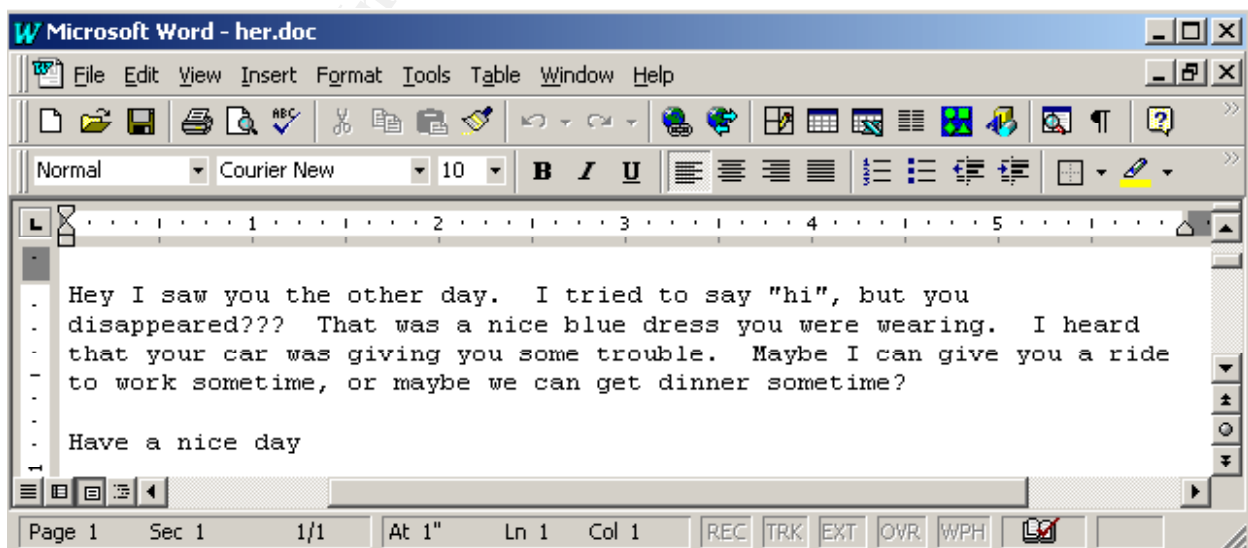


Figure 49 – The her.doc document

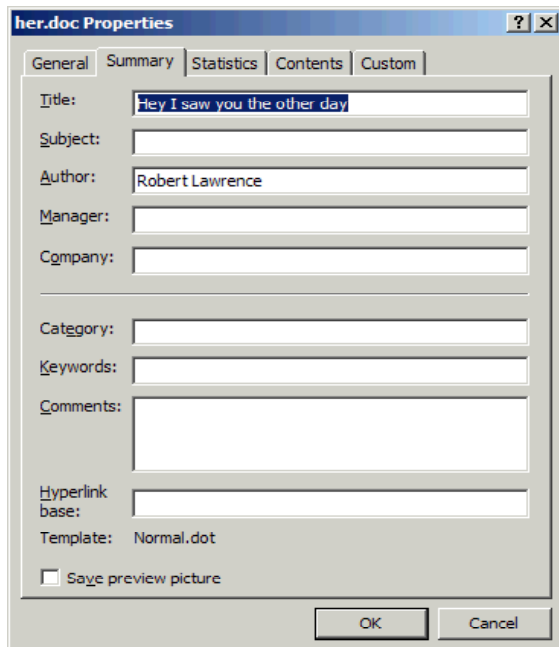


Figure 50 – The properties of the her.doc document

Figure 50 shows that the owner of the “her.doc” file was Robert Lawrence.

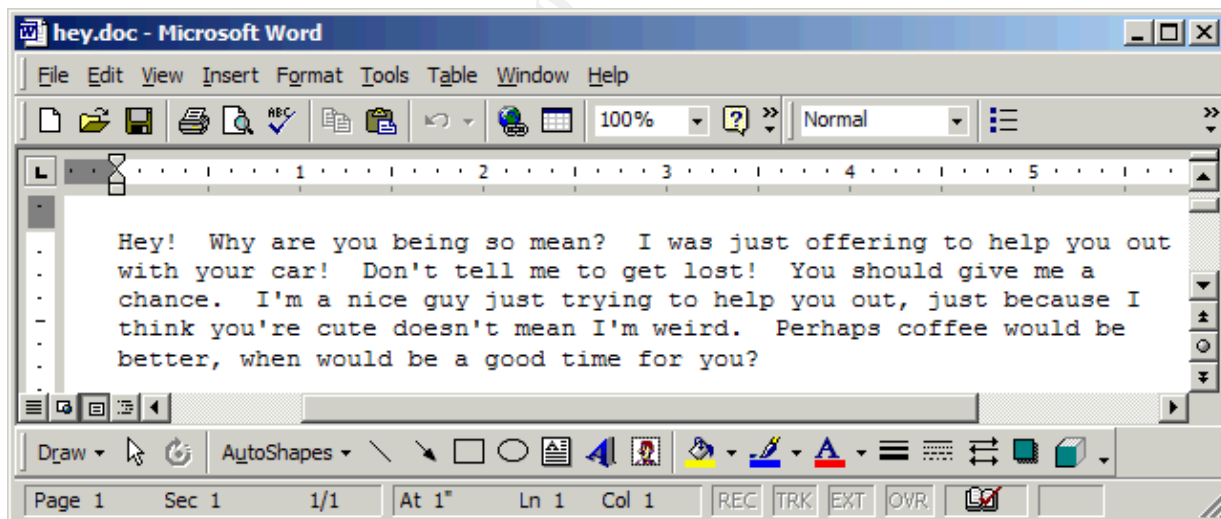


Figure 51 – The hey.doc document

Again, the properties of the “hey.doc” file showed that Robert Lawrence was the owner.



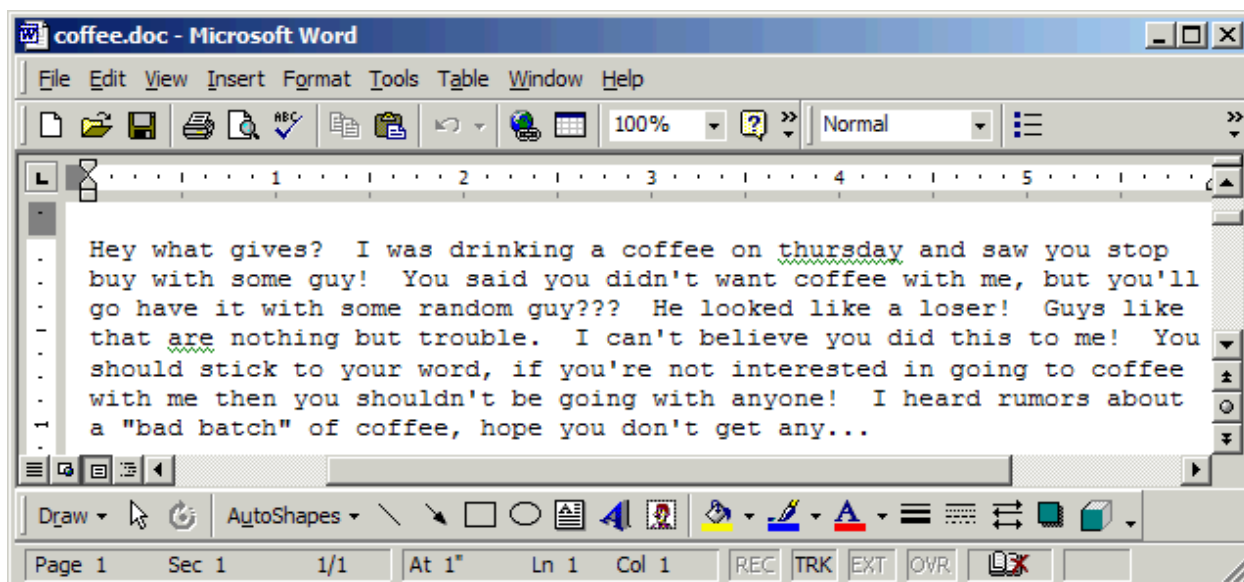


Figure 52 – The coffee.doc document

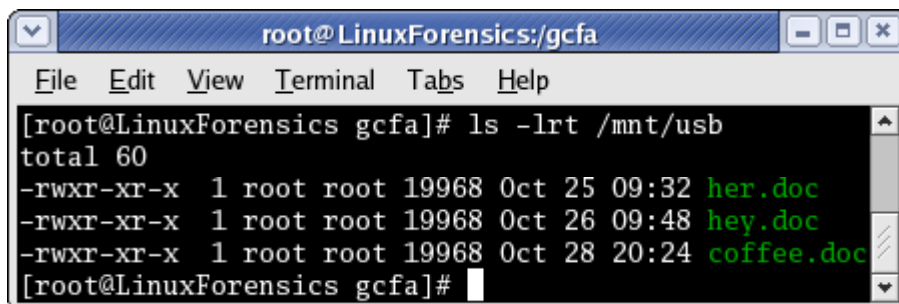
The properties of the “coffee.doc” file also showed that Robert Lawrence was the owner.

## **Conclusions**

To summarize, the examination of the unknown USB image confirmed Ms. Conlay’s harassment claim. The examination showed that Mr. Lawrence had installed a network sniffer (WinDump). He also installed WinPcap, which is a set of libraries that permits WinDump to capture packets. He used the sniffer to capture network packets to and from Ms. Conlay’s PC. This explains how Mr. Lawrence was able to obtain Ms. Conlay’s private email address and how he was able to discover the meeting location and time between Ms. Conlay and her friend. Mr. Lawrence even tried to delete certain files to conceal his actions. The properties of the 3 files that were not deleted show that Mr. Lawrence was the owner. The content of the documents were the emails that Mr. Lawrence had sent to Ms. Conlay and these emails showed that Mr. Lawrence was becoming increasingly aggressive.

## Image Details

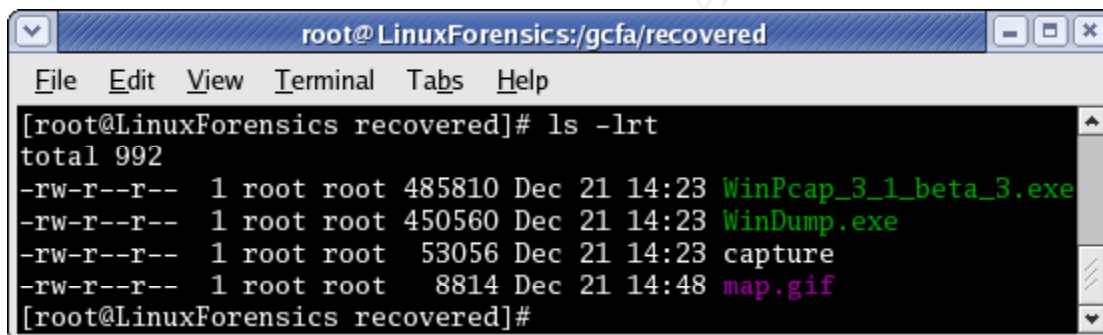
There were three Microsoft Office documents on this USB JumpDrive image. The properties of these documents show that Robert Lawrence was the owner.

A terminal window titled 'root@LinuxForensics:/gcfa' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command '[root@LinuxForensics gcfa]# ls -lrt /mnt/usb' has been executed, resulting in a file listing for the /mnt/usb directory. The listing shows three files: her.doc, hey.doc, and coffee.doc, all with permissions -rwxr-xr-x, owned by root, and sizes of 19968 bytes. The timestamps are Oct 25 09:32, Oct 26 09:48, and Oct 28 20:24 respectively. The prompt is now '[root@LinuxForensics gcfa]#'.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# ls -lrt /mnt/usb
total 60
-rwxr-xr-x 1 root root 19968 Oct 25 09:32 her.doc
-rwxr-xr-x 1 root root 19968 Oct 26 09:48 hey.doc
-rwxr-xr-x 1 root root 19968 Oct 28 20:24 coffee.doc
[root@LinuxForensics gcfa]#
```

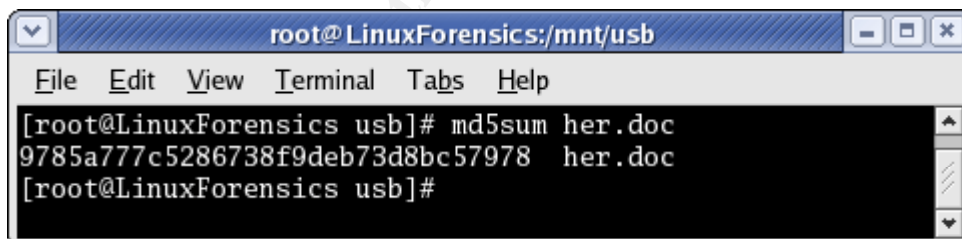
Figure 53 – File listing of the FAT16 image

There were three files that were recovered completely. The fourth file “WinPcap\_3\_1\_beta\_3.exe” had to be rebuilt as it was only partially recovered.

A terminal window titled 'root@LinuxForensics:/gcfa/recovered' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command '[root@LinuxForensics recovered]# ls -lrt' has been executed, resulting in a file listing for the /gcfa/recovered directory. The listing shows four files: WinPcap\_3\_1\_beta\_3.exe, WinDump.exe, capture, and map.gif. The first three files have permissions -rw-r--r-- and sizes of 485810, 450560, and 53056 bytes respectively, all dated Dec 21 14:23. The fourth file, map.gif, has permissions -rw-r--r-- and a size of 8814 bytes, dated Dec 21 14:48. The prompt is now '[root@LinuxForensics recovered]#'.

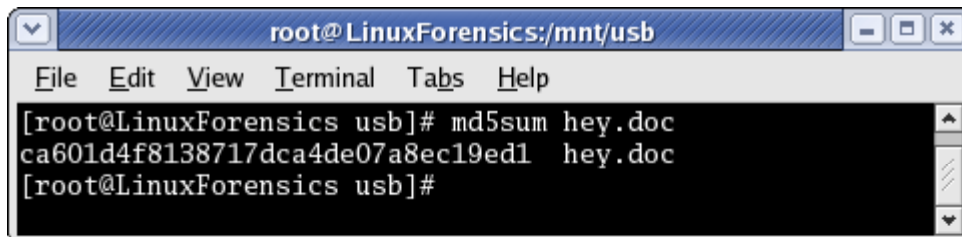
```
root@LinuxForensics:/gcfa/recovered
File Edit View Terminal Tabs Help
[root@LinuxForensics recovered]# ls -lrt
total 992
-rw-r--r-- 1 root root 485810 Dec 21 14:23 WinPcap_3_1_beta_3.exe
-rw-r--r-- 1 root root 450560 Dec 21 14:23 WinDump.exe
-rw-r--r-- 1 root root 53056 Dec 21 14:23 capture
-rw-r--r-- 1 root root 8814 Dec 21 14:48 map.gif
[root@LinuxForensics recovered]#
```

Figure 54 – File listing of the recovered files

A terminal window titled 'root@LinuxForensics:/mnt/usb' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command '[root@LinuxForensics usb]# md5sum her.doc' has been executed, resulting in the MD5 hash '9785a777c5286738f9deb73d8bc57978' for the file her.doc. The prompt is now '[root@LinuxForensics usb]#'.

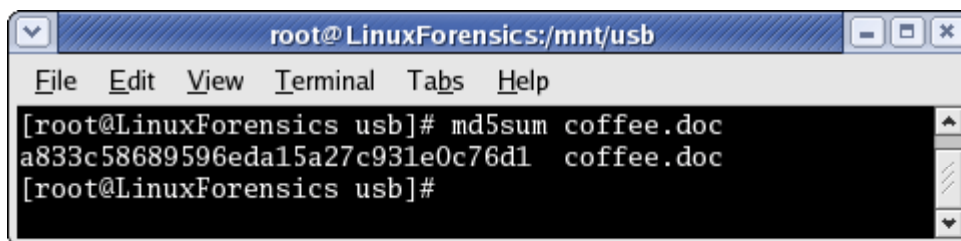
```
root@LinuxForensics:/mnt/usb
File Edit View Terminal Tabs Help
[root@LinuxForensics usb]# md5sum her.doc
9785a777c5286738f9deb73d8bc57978 her.doc
[root@LinuxForensics usb]#
```

Figure 55 – The MD5 hashes of the her.doc file

A terminal window titled 'root@LinuxForensics:/mnt/usb' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The command '[root@LinuxForensics usb]# md5sum hey.doc' has been executed, resulting in the MD5 hash 'ca601d4f8138717dca4de07a8ec19ed1' for the file hey.doc. The prompt is now '[root@LinuxForensics usb]#'.

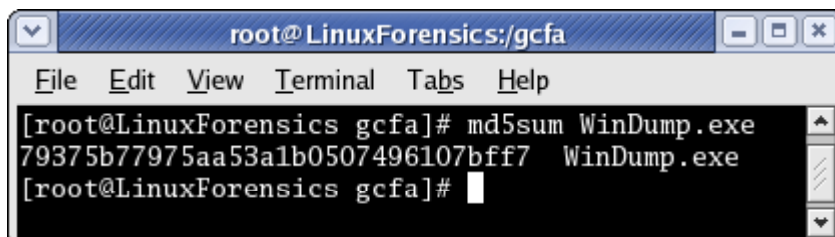
```
root@LinuxForensics:/mnt/usb
File Edit View Terminal Tabs Help
[root@LinuxForensics usb]# md5sum hey.doc
ca601d4f8138717dca4de07a8ec19ed1 hey.doc
[root@LinuxForensics usb]#
```

Figure 56 – The MD5 hashes of the hey.doc file

A terminal window titled 'root@LinuxForensics:/mnt/usb'. The menu bar includes File, Edit, View, Terminal, Tabs, and Help. The terminal text shows the command 'md5sum coffee.doc' being executed, resulting in the output 'a833c58689596eda15a27c931e0c76d1 coffee.doc'.

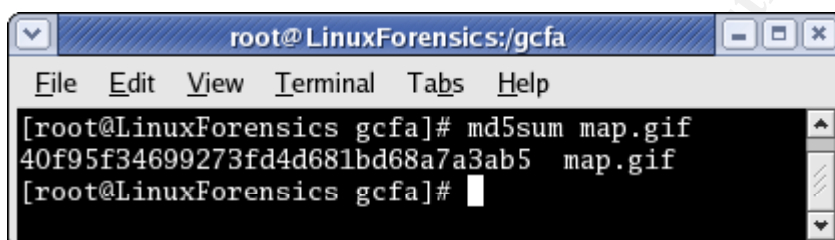
```
root@LinuxForensics:/mnt/usb
File Edit View Terminal Tabs Help
[root@LinuxForensics usb]# md5sum coffee.doc
a833c58689596eda15a27c931e0c76d1 coffee.doc
[root@LinuxForensics usb]#
```

Figure 57 – The MD5 hashes of the coffee.doc file

A terminal window titled 'root@LinuxForensics:/gcfa'. The menu bar includes File, Edit, View, Terminal, Tabs, and Help. The terminal text shows the command 'md5sum WinDump.exe' being executed, resulting in the output '79375b77975aa53a1b0507496107bff7 WinDump.exe'.

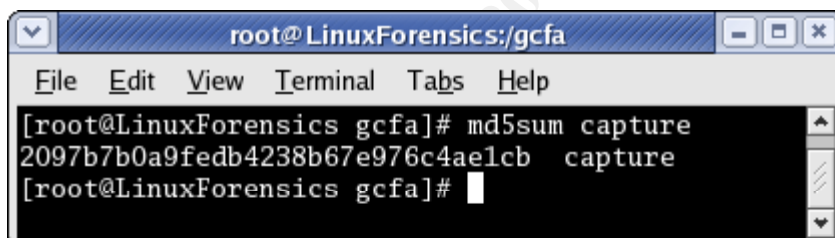
```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# md5sum WinDump.exe
79375b77975aa53a1b0507496107bff7 WinDump.exe
[root@LinuxForensics gcfa]#
```

Figure 58 – The MD5 hashes of the WinDump.exe file

A terminal window titled 'root@LinuxForensics:/gcfa'. The menu bar includes File, Edit, View, Terminal, Tabs, and Help. The terminal text shows the command 'md5sum map.gif' being executed, resulting in the output '40f95f34699273fd4d681bd68a7a3ab5 map.gif'.

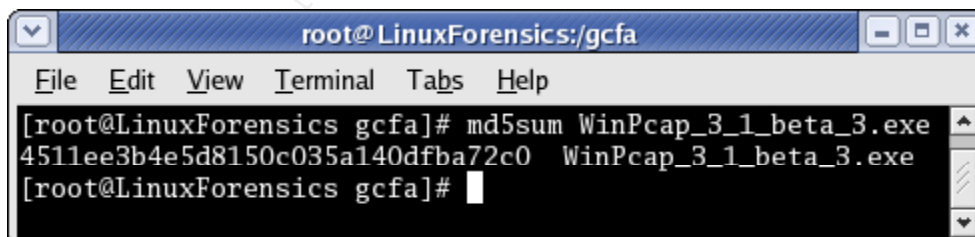
```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# md5sum map.gif
40f95f34699273fd4d681bd68a7a3ab5 map.gif
[root@LinuxForensics gcfa]#
```

Figure 59 – The MD5 hashes of the map.gif file

A terminal window titled 'root@LinuxForensics:/gcfa'. The menu bar includes File, Edit, View, Terminal, Tabs, and Help. The terminal text shows the command 'md5sum capture' being executed, resulting in the output '2097b7b0a9fedb4238b67e976c4ae1cb capture'.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# md5sum capture
2097b7b0a9fedb4238b67e976c4ae1cb capture
[root@LinuxForensics gcfa]#
```

Figure 60 – The MD5 hashes of the capture file

A terminal window titled 'root@LinuxForensics:/gcfa'. The menu bar includes File, Edit, View, Terminal, Tabs, and Help. The terminal text shows the command 'md5sum WinPcap\_3\_1\_beta\_3.exe' being executed, resulting in the output '4511ee3b4e5d8150c035a140dfba72c0 WinPcap\_3\_1\_beta\_3.exe'.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# md5sum WinPcap_3_1_beta_3.exe
4511ee3b4e5d8150c035a140dfba72c0 WinPcap_3_1_beta_3.exe
[root@LinuxForensics gcfa]#
```

Figure 61 – The MD5 hashes of the WinPcap\_3\_1\_beta\_3.exe file

```

root@LinuxForensics:gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# cat timeline.mac
Mon Oct 25 2004 00:00:00 19968 .a. -/-RWXTRWXTWX 0 0 3 /her.doc
Mon Oct 25 2004 08:32:06 19968 ..c -/-RWXTRWXTWX 0 0 3 /her.doc
Mon Oct 25 2004 08:32:08 19968 m.. -/-RWXTRWXTWX 0 0 3 /her.doc
Tue Oct 26 2004 00:00:00 19968 .a. -/-RWXTRWXTWX 0 0 4 /hey.doc
Tue Oct 26 2004 08:48:06 19968 ..c -/-RWXTRWXTWX 0 0 4 /hey.doc
Tue Oct 26 2004 08:48:10 19968 m.. -/-RWXTRWXTWX 0 0 4 /hey.doc
Wed Oct 27 2004 00:00:00 485810 .a. -/-RWXTRWXTWX 0 0 7 /WinPcap_3_1_beta_3.exe (_I
NPCA~1.EXE) (deleted)
450560 .a. -/-RWXTRWXTWX 0 0 12 /WinDump.exe (_INDUMP.EXE)
(deleted)
Wed Oct 27 2004 16:23:50 485810 m.. -/-RWXTRWXTWX 0 0 10 /WinPcap_3_1_beta_3.exe (_I
NPCA~1.EXE) (deleted)
Wed Oct 27 2004 16:23:54 485810 ..c -/-RWXTRWXTWX 0 0 7 /WinPcap_3_1_beta_3.exe (_I
NPCA~1.EXE) (deleted)
485810 ..c -/-RWXTRWXTWX 0 0 10 /WinPcap_3_1_beta_3.exe (_I
NPCA~1.EXE) (deleted)
Wed Oct 27 2004 16:23:56 485810 m.. -/-RWXTRWXTWX 0 0 7 /WinPcap_3_1_beta_3.exe (_I
NPCA~1.EXE) (deleted)
Wed Oct 27 2004 16:24:02 450560 m.. -/-RWXTRWXTWX 0 0 14 /WinDump.exe (_INDUMP.EXE)
(deleted)
Wed Oct 27 2004 16:24:04 450560 ..c -/-RWXTRWXTWX 0 0 12 /WinDump.exe (_INDUMP.EXE)
(deleted)
450560 ..c -/-RWXTRWXTWX 0 0 14 /WinDump.exe (_INDUMP.EXE)
(deleted)
Wed Oct 27 2004 16:24:06 450560 m.. -/-RWXTRWXTWX 0 0 12 /WinDump.exe (_INDUMP.EXE)
(deleted)
Thu Oct 28 2004 00:00:00 8814 .a. -/-RWXTRWXTWX 0 0 16 /_ap.gif (deleted)
485810 .a. -/-RWXTRWXTWX 0 0 10 /WinPcap_3_1_beta_3.exe (_I
NPCA~1.EXE) (deleted)
53056 .a. -/-RWXTRWXTWX 0 0 15 /_apture (deleted)
8814 .a. -/-RWXTRWXTWX 0 0 17 /_ap.gif (deleted)
19968 .a. -/-RWXTRWXTWX 0 0 18 /coffee.doc
450560 .a. -/-RWXTRWXTWX 0 0 14 /WinDump.exe (_INDUMP.EXE)
(deleted)
Thu Oct 28 2004 11:08:24 53056 ..c -/-RWXTRWXTWX 0 0 15 /_apture (deleted)
Thu Oct 28 2004 11:11:00 53056 m.. -/-RWXTRWXTWX 0 0 15 /_apture (deleted)
Thu Oct 28 2004 11:17:44 8814 ..c -/-RWXTRWXTWX 0 0 16 /_ap.gif (deleted)
8814 ..c -/-RWXTRWXTWX 0 0 17 /_ap.gif (deleted)
Thu Oct 28 2004 11:17:46 8814 m.. -/-RWXTRWXTWX 0 0 16 /_ap.gif (deleted)
8814 m.. -/-RWXTRWXTWX 0 0 17 /_ap.gif (deleted)
Thu Oct 28 2004 19:24:46 19968 ..c -/-RWXTRWXTWX 0 0 18 /coffee.doc
Thu Oct 28 2004 19:24:48 19968 m.. -/-RWXTRWXTWX 0 0 18 /coffee.doc
[root@LinuxForensics gcfa]#

```

Figure 62 – Timeline of the FAT 16 image

© SANS Institute

The following is the screen capture of the dirty word list.

```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# cat dirty-word-list
Timeline:
WinDump.exe
WinPcap_3_1_beta_3.exe

Capture:
flowergirl96
Hollywood
McCa

WinDump.exe
/tcpdump/master/tcpdump/addrtoname.c,v 1.96.2.6 2004/03/24 04:14
:31 guy Exp $ (LBL)
windump
icmp6: router advertisement
%s: pcap_loop: %s
%d packets received by filter
%u packets captured
%s version %s, based on tcpdump version %s

WinPcap_3_1_beta_3.exe:
OLEAUT32.DLL
data\Main\0\npf.sys
data\Main\1\npf.sys
data\Main\2\packet.dll
data\Main\3\wanpacket.dll
data\Main\4\packet.dll
data\Main\5\packet.dll
data\Main\6\npf.vxd
data\Main\7\wpcap.dll

[root@LinuxForensics gcfa]#
```

These key words were obtained from the Timeline.

These key words were obtained from the network capture.

These key words were obtained from the WinDump.exe program.

These key words were obtained from the WinPcap\_3\_1\_beta\_3.exe program.

Figure 63 – The dirty word list

## Forensic Details

The investigation revealed that Mr. Lawrence used the “WinDump.exe” 3.8.3 beta and the “WinPcap\_3\_1\_beta\_3.exe” executable programs.

The “WinDump.exe” is a network wiretap program that captures network packets on a network segment and saves this capture to a file. Therefore, if the usernames, passwords, email conversation, etc are transmitted across a network segment, the WinDump.exe will be able to capture all this information. The “WinDump.exe” program is based on the Linux version of **tcpdump** utility. This program was accessed on Oct 27 2004 at 00:00:00 and again on Oct 28 2004 at 00:00:00.

The “WinPcap\_3\_1\_beta\_3.exe” is an executable program that installs the libraries that are required by WinDump.exe to capture network packets. “It includes a kernel-level packet filter, a low-level dynamic link library (packet.dll), and a high-level and system-independent library (wpcap.dll, based on libpcap version 0.6.2). The packet filter is a device driver that adds to Windows 95, 98, ME, 2000, and XP the ability to capture and send raw data from a network card, with the possibility to filter and store in a buffer the captured packets. Packet.dll is an API that can be used to directly access the functions of the packet driver, offering a programming interface independent from the Microsoft OS. Wpcap.dll exports a set of high level captures primitives that are compatible with libpcap, the well-known Unix capture library. These functions allow to capture packets in a way independent from the underlying network hardware and operating system” [4].

Mr. Lawrence first used the “WinPcap\_3\_1\_beta\_3.exe” to install the libraries that were required by WinDump. He then used the “WinDump.exe” program to capture the network activities initiated by Ms. Conlay. This was evident as there was a WinDump capture file with information on a particular network session between Ms. Conlay’s PC and [www.hotmail.com](http://www.hotmail.com) (an email service). Mr. Lawrence was able to analyze this WinDump capture file and find Ms. Conlay’s private email address, and the meeting time and location with her friend. Mr. Lawrence then used <http://www.mapblast.com> to determine the meeting location. It was also apparent that Mr. Lawrence knew that he had violated his company’s policies because he deleted the capture file, the WinDump and WinPCap programs once he had finished analyzing the capture file. This was probably done in an attempt to conceal his activities.

The “WinDump.exe”, “WinPcap\_3\_1\_beta\_3.exe”, WinDump capture file, and map of the meeting location were recovered using the following methods:

- The FAT16 logical partition was extracted using the **dcfldd** utility.
- The **fls** and the **mactime** utilities were used to gather the timeline of this FAT16 logical partition. The timeline showed the four deleted files (WinDump.exe, WinPcap\_3\_1\_beta\_3.exe, \_ap.gif and \_apture) and their associated inode addresses. The timeline also indicated when the “WinDump.exe” and the “WinPcap\_3\_1\_beta\_3.exe” were last run.
- The four files were deleted, thus their inode addresses were changed from



allocated to not allocated. However, the actual data associated with the files were still on the partition. The **istat** utility was then used to extract inode information such as the file name, file size and the blocks that the file occupies. With this information, the **dcfldd** utility was used to extract the four deleted files.

The Google search engine was used to locate both the “WinDump.exe” and the “WinPcap\_3\_1\_beta\_3.exe” files.

The “WinDump.exe” was found at <http://windump.polito.it/install/default.htm>. The **md5sum** utility was used to calculate the MD5 hash of this downloaded file. The MD5 hash was then compared to the MD5 of the recovered WinDump.exe file. The MD5 hashes matched. Therefore, it was concluded that the recovered file was the WinDump 3.8.3 beta.

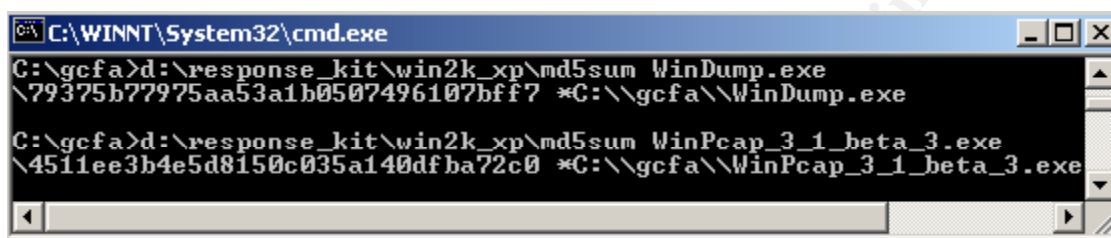
The “WinPcap\_3\_1\_beta\_3.exe” was found at <http://www.oltenia.ro/download/pub/windows/network%20tools/ethereal/>. Again the **md5sum** utility was used to calculate the MD5 hash of this downloaded file. Since the “WinPcap\_3\_1\_beta\_3.exe” could only be partially recovered, the MD5 hash of the downloaded file could not be used at this moment to match the MD5 hash of the partially recovered file. Thus, an attempt to restore the partially recovered file into the complete file was initiated. Both the downloaded and partially recovered WinPcap\_3\_1\_beta\_3.exe files were analyzed using the **khxedit** utility. Approximately 20% of the data was missing from the partially recovered file when compared to the downloaded file. Therefore, the missing data was copied from the downloaded file to the partially recovered file. Once the extra bits at the end of the partially recovered file were removed, the **md5sum** utility was used to calculate the MD5 hash of this file. The MD5 hash matched the MD5 hash of the downloaded “WinPcap\_3\_1\_beta\_3.exe” file. Therefore, it was concluded that the partially recovered file was the WinPcap version 3.1 beta 3.

The “\_apture” file was also recovered. The **file** utility indicated that this was a tcpdump capture file. The **ethereal** utility was used to analyze this capture file. Further investigation on this capture file revealed the personal email address of Ms. Conlay, her friend and their meeting time and location. It was indicated in Ms. Conlay’s email that the coffee shop was at an out of the way location. Since Mr. Lawrence had appeared unexpectedly at the meeting location, it was concluded that he knew about the meeting time and location. At this point, it was fairly conclusive that Mr. Lawrence was the person who downloaded the WinDump and the WinPcap programs. He then ran these programs and analyzed the WinDump captured file in order to determine where and when Ms. Conlay was meeting her friend. This also explains how Mr. Lawrence was able to obtain Ms. Conlay’s personal email address.

In order to analyze both the WinPcap and the WinDump programs, a VMware session of Windows 2000 Professional SP2 was used. The IP address of 192.168.2.1 (the gateway) was assigned to the VMware and 192.168.2.10 was assigned to the Windows 2000 Professional SP2. VMware is a program that allows different virtual

operating systems to exist within an operating system. Thus, a session of Windows 2000 can be run on the Fedora Core 2 Linux operating system. The VMware session can be configured such that any activities executed in that session will not affect the main operating system. This is very useful in forensic cases that involve analyzing unknown programs or potentially malicious programs.

Both the WinPcap and WinDump executable programs were copied to the Windows 2000 VMware session via a floppy to the C:\gcfa folder. The SANS Track 8 Course CD, which contains the Windows forensic tools, was mounted on the Windows 2000 VMware session on the D:\ drive. The **md5sum** utility on the SANS CD was used to verify the MD5 hashes of both the WinPcap and WinDump programs.



```
C:\WINNT\System32\cmd.exe
C:\gcfa>d:\response_kit\win2k_xp\md5sum WinDump.exe
\79375b77975aa53a1b0507496107bfff7 *C:\gcfa\WinDump.exe

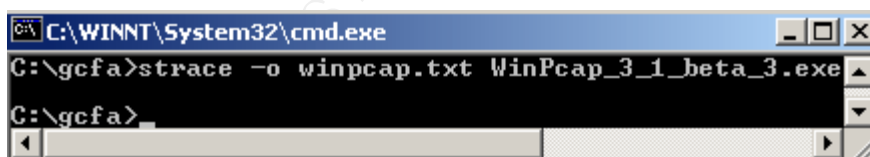
C:\gcfa>d:\response_kit\win2k_xp\md5sum WinPcap_3_1_beta_3.exe
\4511ee3b4e5d8150c035a140dfba72c0 *C:\gcfa\WinPcap_3_1_beta_3.exe
```

Figure 64 – MD5 Hash of the WinDump and WinPcap programs

### WinPcap 3 1 beta 3.exe Program Analysis

The WinPcap\_3\_1\_beta\_3 program contains the libraries that WinDump requires. Therefore, it must be installed first. The **strace** utility was used to monitor the WinPcap\_3\_1\_beta\_3 installation. **Strace** is a utility that monitors the interactions of a program and the operating system during the program's run-time process and writes the interactions to a trace file. A copy of the **strace** utility was downloaded from [http://www.bindview.com/Support/RAZOR/Utilities/Windows/strace\\_readme.cfm](http://www.bindview.com/Support/RAZOR/Utilities/Windows/strace_readme.cfm) and installed on the C:\gcfa folder. The **strace** utility was run with the following option:

- **-o output.txt** – This option invokes the **strace** utility to write the trace to the "output.txt" file.



```
C:\WINNT\System32\cmd.exe
C:\gcfa>strace -o winpcap.txt WinPcap_3_1_beta_3.exe
C:\gcfa>
```

Figure 65 – Strace output for WinPcap installation



```

C:\WINNT\System32\cmd.exe
C:\gcfa>dir winpcap.txt
Volume in drive C has no label.
Volume Serial Number is 942D-D5AE

Directory of C:\gcfa

12/31/2004  10:26a                2,011,831 winpcap.txt
               1 File(s)                2,011,831 bytes
               0 Dir(s)          3,309,809,664 bytes free

C:\gcfa>

```

Figure 66 – Strace output “winpcap.txt” file

The notepad tool was used to analyze the strace output file “winpcap.txt”. The strace output “winpcap.txt” file is long, therefore only the pertinent sections of the strace output file will be displayed and analyzed.

No.	WinPcap Strace Output
3701	<b>NtCreateFile</b> (0xc0100080, {24, 0, 0x40, 0, 1244476, "???\C:\DOCUME~1\WINFOR~1\LOCALS~1\Temp\1P743063\WinPcap_3_1_beta_3\splash.bmp"}, 0x0, 128, 1, 1, 96, 0, 0, ... 292, {status=0x0, info=1}, ) == 0x0
3702	<b>NtSetInformationFile</b> (292, 1244532, 40, <b>Basic</b> , ... {status=0x0, info=0}, ) == 0x0
3703	<b>NtClose</b> (292, ... ) == 0x0

Table 2 – WinPcap.txt Strace Output #1

The definition of the **NtCreateFile**, **NtSetInformationFile**, and the **NtClose** functions can be found in the MSDN Library. The definitions are listed below:

“The **NtCreateFile** function either creates a new file or directory, or opens an existing file, device, directory, or volume...**NtCreateFile** is equivalent to the **ZwCreateFile** function documented in the DDK” [6][7].

The **NtSetInformationFile** function is equivalent to the **ZwSetInformationFile** function. “The **ZwSetInformationFile** routine changes various kinds of information about a file object” [8]. In Table 2, if the MAC times of a file are changed, the **NtSetInformationFile** with the **Basic** value will be set. If a file is to be deleted, the **NtSetInformationFile** with the **Disposition** value will be set and the file will be deleted once it is closed with the **NtClose** function.

The “winpcap.txt” strace output file contained many temporarily files and folders that were created during the WinPcap installation but only a few of these files were analyzed.

The strace output of the “winpcap.txt” in Table 2 indicates that the **NtCreateFile** function was first used to create the temporarily file “splash.bmp”. The

**NtSetInformationFile** with the **Basic** value was used next to indicate the MAC times of this file were modified. The **NtClose** function was used to end the file operation.

No.	WinPcap Strace Output
14215	<b>NtCreateFile</b> (0xc0100080, {24, 0, 0x40, 0, 17497612, "???\C:\WINNT\System32\packet.dll"}, 0x0, 128, 1, 1, 96, 0, 0, ... 328, {status=0x0, info=1}, ) == 0x0
14216	<b>NtSetInformationFile</b> (328, 17497668, 40, Basic, ... {status=0x0, info=0}, ) == 0x0
14217	<b>NtClose</b> (328, ... ) == 0x0

Table 3 – WinPcap.txt Strace Output #2

Table 3 indicates that the “packet.dll” file was created in the folder C:\WINNT\System32. Again, the **NtSetInformationFile** with the **Basic** value was used next to indicate the MAC times of this file was modified and the **NtClose** function was used to end the file operation.

No.	WinPcap Strace Output
20173	<b>NtOpenFile</b> (0x10080, {24, 0, 0x40, 0, 0, "???\C:\DOCUME~1\WINFOR~1\LOCALS~1\Temp\1P743063\WinPcap_3_1_beta_3\splash.bmp"}, 7, 2113600, ... 284, {status=0x0, info=1}, ) == 0x0
20174	<b>NtQueryInformationFile</b> (284, 1244160, 8, AttributeFlag, ... {status=0x0, info=8}, ) == 0x0
20175	<b>NtSetInformationFile</b> (284, 1244208, 1, Disposition, ... {status=0x0, info=0}, ) == 0x0
20176	<b>NtClose</b> (284, ... ) == 0x0

Table 4 – WinPcap.txt Strace Output #3

Table 4 introduces the functions **NtOpenFile** and **NtQueryInformationFile**. The **NtOpenFile** function indicated that a particular file is open for file operation [9]. In Table 4, the **NtOpenFile** function opened the “splash.bmp” file. The **NtQueryInformationFile** with the AttributeFlag value returns the attribute of the file, eg if the file is hidden, read-only, system [10]. The next function **NtSetInformationFile** with the **Disposition** value indicated that this file that will be deleted once the file is closed. The “splash.bmp” file was deleted after the **NtClose** function.

The complete analysis of the WinPCap executable program indicates the following:

- The npf.sys, packet.dll, wanpacket.dll, wpcap.dll and pthreadVC.dll were installed in the C:\WINNT\System32 folder.
- The npf\_mgm.exe, daemon\_mgm.exe, rpcapd.exe, NetMonInstaller.exe, Uninstall.exe, and Install.log were installed in the C:\Program Files\WinPcap folder.

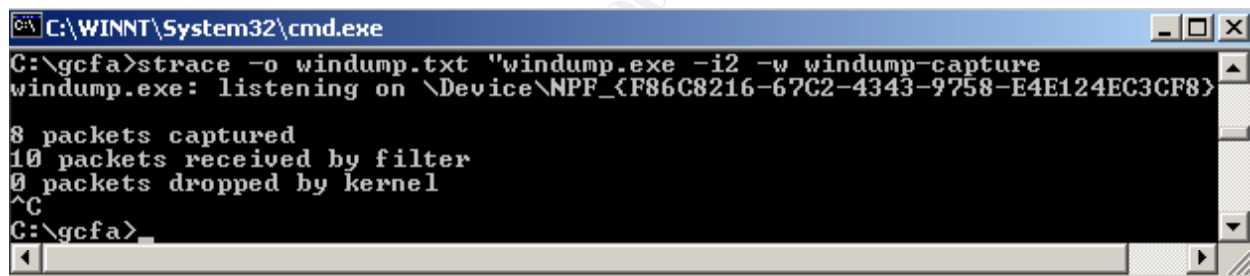
## WinDump.exe Program Analysis

Since the MD5 hash of the recovered WinDump program matched the MD5 hash of the downloaded WinDump program from <http://windump.polito.it/install/default.htm>, this program was classified as a known program. Thus, prior to any program analysis, the function of this program was already known. However, the methodology used for analyzing an unknown program would consist of using the **strace** utility to:

- monitor what files are installed on the system during the program execution.
- monitor when these files are used during the program execution
- monitor the registry keys that are installed during the program execution
- monitor the network connections the unknown program may have initiated during the program execution.

The **strace** utility was used to monitor the WinDump program with the following options (Figure 67):

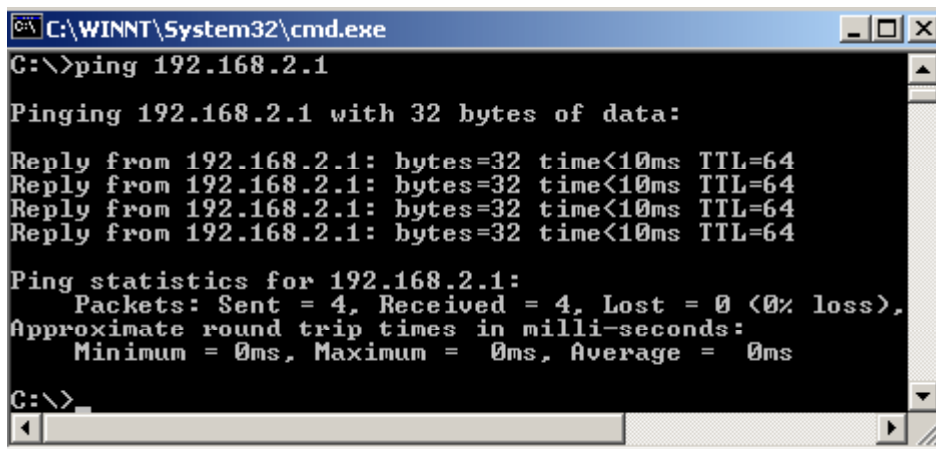
- -i2 – This option invokes the WinDump program to use the active network interface on the operating system.
- -w windump-capture – This option invokes the WinDump program to write the output to the “windump-capture” file.



```
C:\WINNT\System32\cmd.exe
C:\gcfa>strace -o windump.txt "windump.exe -i2 -w windump-capture"
windump.exe: listening on \Device\NPF_{F86C8216-67C2-4343-9758-E4E124EC3CF8}
8 packets captured
10 packets received by filter
0 packets dropped by kernel
^C
C:\gcfa>
```

Figure 67 – Strace output for WinDump capture

Once the WinDump program was executed, a PING to 192.168.2.1 (the gateway) was initiated (Figure 68). This was done to generate network packets so that WinDump can capture these packets.



```
C:\WINNT\System32\cmd.exe
C:\>ping 192.168.2.1

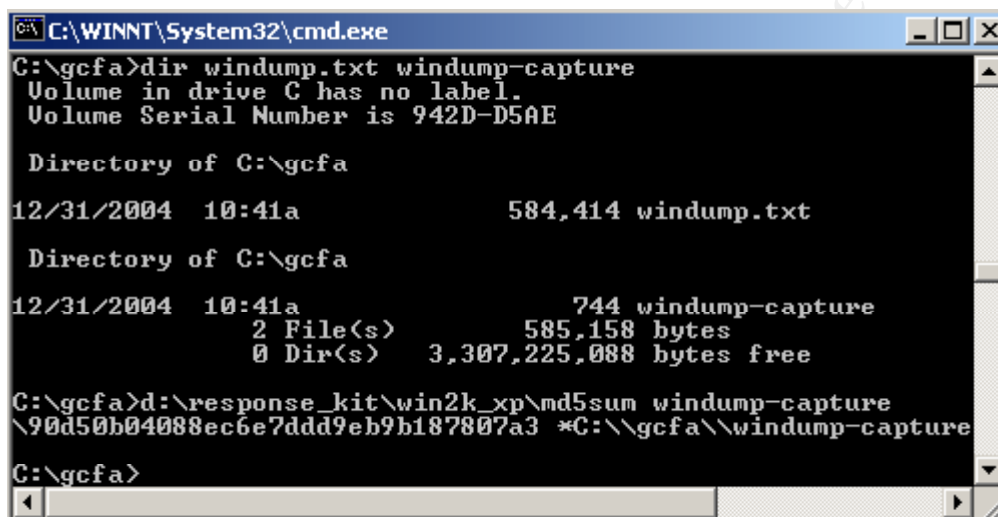
Pinging 192.168.2.1 with 32 bytes of data:

Reply from 192.168.2.1: bytes=32 time<10ms TTL=64
Reply from 192.168.2.1: bytes=32 time<10ms TTL=64
Reply from 192.168.2.1: bytes=32 time<10ms TTL=64
Reply from 192.168.2.1: bytes=32 time<10ms TTL=64

Ping statistics for 192.168.2.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

Figure 68 – Ping 192.168.2.1 (Gateway)



```
C:\WINNT\System32\cmd.exe
C:\gcfa>dir windump.txt windump-capture
Volume in drive C has no label.
Volume Serial Number is 942D-D5AE

Directory of C:\gcfa
12/31/2004  10:41a                584,414 windump.txt

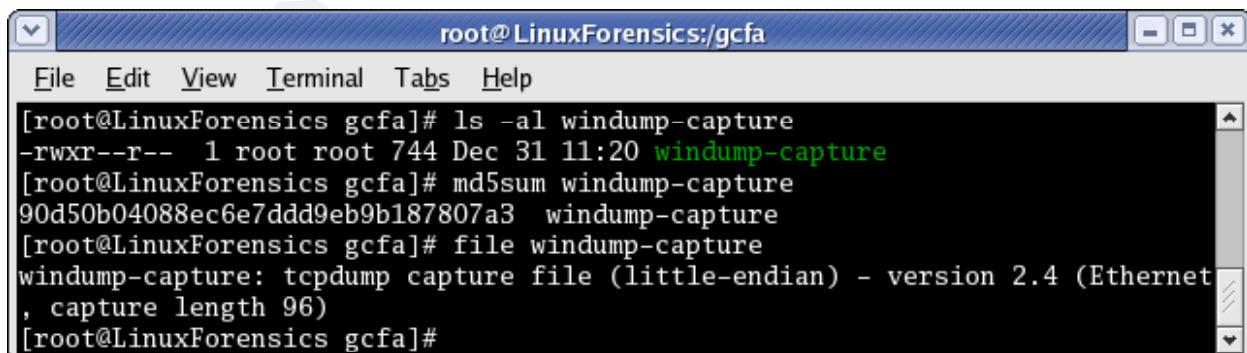
Directory of C:\gcfa
12/31/2004  10:41a                744 windump-capture
                2 File(s)          585,158 bytes
                0 Dir(s)      3,307,225,088 bytes free

C:\gcfa>d:\response_kit\win2k_xp\md5sum windump-capture
90d50b04088ec6e7ddd9eb9b187807a3 *C:\gcfa\windump-capture

C:\gcfa>
```

Figure 69 – MD5 hash of the windump-capture file

Figure 69 shows the **strace** output file “windump.txt”, WinDump capture file “windump-capture” and the MD5 hash of the “windump-capture” file. The “windump-capture” file was copied to the Linux Forensics workstation /gcfa folder via a floppy disk.



```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# ls -al windump-capture
-rwxr--r--  1 root root 744 Dec 31 11:20 windump-capture
[root@LinuxForensics gcfa]# md5sum windump-capture
90d50b04088ec6e7ddd9eb9b187807a3  windump-capture
[root@LinuxForensics gcfa]# file windump-capture
windump-capture: tcpdump capture file (little-endian) - version 2.4 (Ethernet
, capture length 96)
[root@LinuxForensics gcfa]#
```

Figure 70 – MD5 verification of the windump-capture file

The **md5sum** utility was used to verify the MD5 hash of the “windump-capture” file. The **file** utility was used next to verify that the “windump-capture” was a tcpdump capture file. The **ethereal** utility was used next to examine this “windump-capture” file.

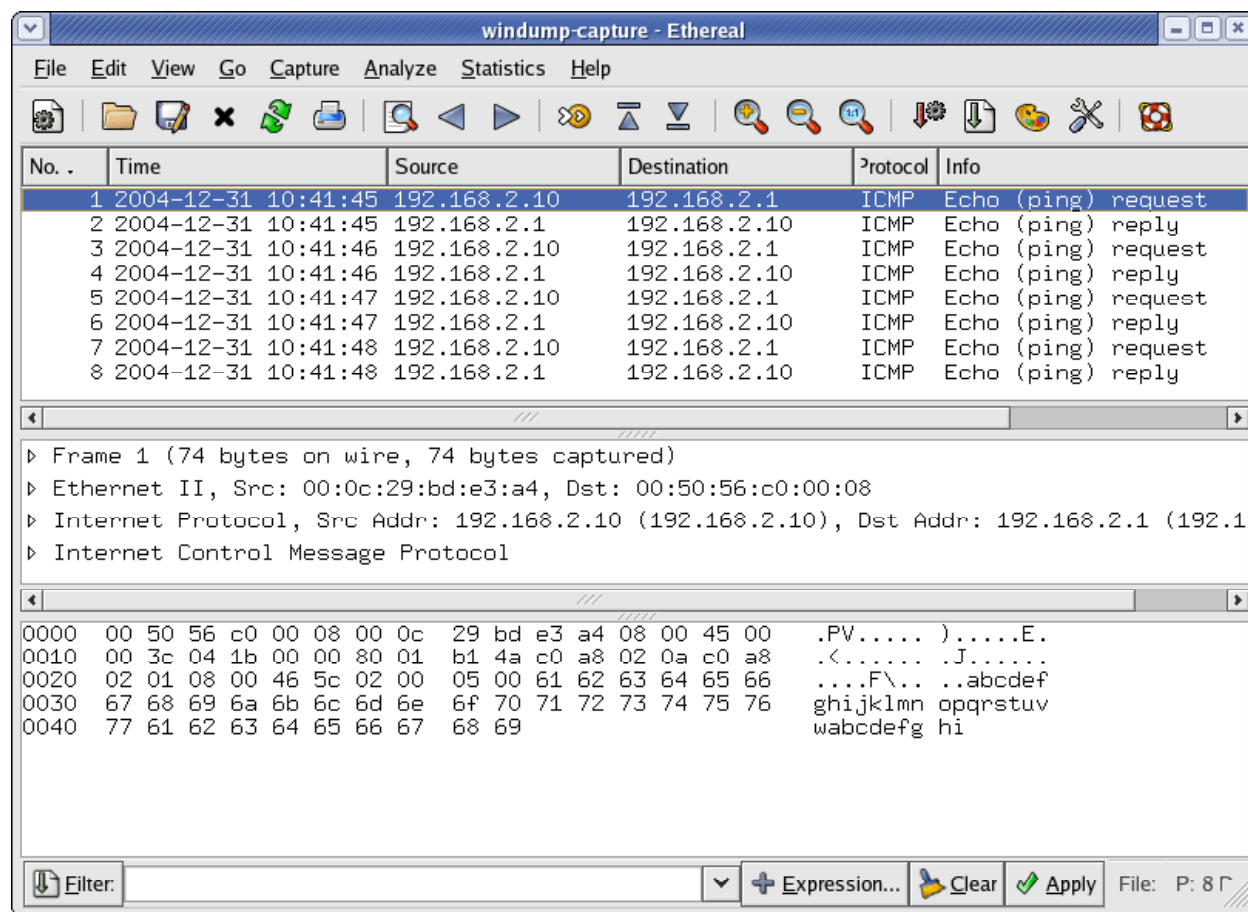


Figure 71 – Ethereal of the windump-capture file

Figure 71 shows 4 ICMP Echo requests (from 192.168.2.10 to 192.168.2.1) and 4 ICMP Echo replies (from 192.168.2.1 to 192.168.2.10). This result was expected as there were 4 PING replies from 192.168.2.1 (showed in Figure 68).

The notepad tool was used to analyze the strace output file “windump.txt”. The pertinent section of the strace output file “windump.txt” will be displayed and analyzed.

No.	“WinDump.txt” Strace Output
97	NtQueryAttributesFile ({24, 0, 0x40, 0, 0, “\\??\\C:\\WINNT\\System32\\wpcap.dll”}, 1243076, ... ) == 0x0
100	NtOpenFile (0x100020, {24, 0, 0x40, 0, 0, “\\??\\C:\\WINNT\\System32\\wpcap.dll”}, 5, 96, ... 32, {status=0x0, info=1}, ) == 0x0
...	
117	NtQueryAttributesFile ({24, 0, 0x40, 0, 0, “\\??\\C:\\WINNT\\System32\\packet.dll”}, 1242952, ... ) == 0x0

120	NtOpenFile (0x100020, {24, 0, 0x40, 0, 0, \"\\?\\C:\\WINNT\\System32\\packet.dll\"}, 5, 96, ... 28, {status=0x0, info=1}, ) == 0x0
...	
144	NtQueryAttributesFile ({24, 0, 0x40, 0, 0, \"\\?\\C:\\WINNT\\System32\\WanPacket.dll\"}, 1242828, ... ) == 0x0
147	NtOpenFile (0x100020, {24, 0, 0x40, 0, 0, \"\\?\\C:\\WINNT\\System32\\WanPacket.dll\"}, 5, 96, ... 32, {status=0x0, info=1}, ) == 0x0

Table 5 – Strace output for WinDump program #1

Table 5 indicates the following:

- The WinDump program first checked if the “wpcap.dll”, “packet.dll” and “WanPacket.dll” files exist through the use of the **NtFsQueryAttributesFile** function. This function determines if a file exists on a file system. This function is often used because it does not change the last accessed time on the file [11].
- The “wpcap.dll”, “packet.dll” and “WanPacket.dll” files are the WinPCap library files. These files were installed in the folder “C:\\WINNT\\System32” during the WinPcap installation process.
- Once these files were located, the strace output showed that the WinDump program open these files. This established the WinDump’s reliance on the WinPcap libraries.

The following libraries were found throughout the strace output “windump.txt”. These libraries were opened using the **NtOpenFile** function.

- iphlapi.dll – This is the IP Helper API. It is used to retrieve network configuration settings on a Windows operating system [12].
- icmp.dll – This library is used by the PING utility [13]. The PING utility is used to determine if a host is on the network.
- mprapi.dll – This library is used to configure the Microsoft Windows 2000 routers [14].
- samlib.dll – This library is used for the Security Authority Manager API [15].
- netapi32.dll – This library is used by applications to access a Microsoft network [16].
- secur32.dll – This library contains Windows Security functions [17].
- netrap.dll – This is the Net Remote Admin Protocol dynamic link library and this library is part of the installation of the Windows 2000 operating system [18].
- dnsapi.dll – This is the Domain Name Server API dynamic link library. It is used by DNS resolvers to initiate DNS name lookups [19].
- activeds.dll – This is the Active Directory Services Interfaces (ADSI) API dynamic link library [20].
- adslrpc.dll – This is one of the Active Directory Services Interfaces dynamic link libraries [21].
- rtutils.dll – This is the routing utilities. It is responsible for generating diagnostics for the Windows Routing and Remote Access Service components [22].
- setupapi.dll – This library is used by installers and setup applications [23].
- userenv.dll – This library is used to create and manage user profiles [24].

- rasapi32.dll – This library is used to control modem connections [25].
- rasman.dll – This is the Remote Access Server Manager DLL. It is used by applications to initiate remote access functions [26].
- tapi32.dll – This is the Microsoft Windows Telephony API client DLL [27].
- dhcpcsvc.dll – This is the DHCP Client service DLL. This library is used to provide DHCP client services [28].
- clbcatq.dll – This library is part of the Windows COM services [29].

No.	"WinDump.txt" Strace Output
3917	NtCreateFile ... 428, {status=0x0, info=0}, ) == 0x0
3919	NtDeviceIoControlFile (428, 0, 0x0, 0x0, 0x1cf7, 0x0, 0, 26, ... {status=0x0, info=26}, "N\0P\0F\00\00\00\00\00\00\00\00\03\06\0", ) == 0x0
3921	NtDeviceIoControlFile (428, 0, 0x0, 0x0, 0x80000004, "\5\1\1\0\4\0\0\0\0\0\0\0\0\0\0\0", 15, 15, ... {status=0x0, info=15}, "\5\1\1\0\4\0\0\0\0\0\2\0\0\0\0\0", ) == 0x0
3922	NtDeviceIoControlFile (428, 0, 0x0, 0x0, 0x80000000, "\17\1\1\0\4\0\0\0\0\2\0\0\0\0\0", 15, 15, ... {status=0x0, info=15}, "\17\1\1\0\4\0\0\0\0\2\0\0\0\0\0", ) == 0x0
3925	NtDeviceIoControlFile (428, 0, 0x0, 0x0, 0x80000000, "\16\1\1\0\4\0\0\0\0\0\0\0\0\0\0", 15, 15, ... {status=0x0, info=15}, "\16\1\1\0\4\0\0\0\0\0\0\0\0\0\0", ) == 0x0
3928	NtDeviceIoControlFile (428, 0, 0x0, 0x0, 0x2578, "@B\17\0", 4, 0, ... {status=0x0, info=1000000}, 0x0, ) == 0x0
3929	NtDeviceIoControlFile (428, 0, 0x0, 0x0, 0x1cf6, "\200>\0\0", 4, 0, ... {status=0x0, info=16000}, 0x0, ) == 0x0
3930	NtDeviceIoControlFile (428, 0, 0x0, 0x0, 0x1cf8, "\377\377\377\377", 4, 0, ... {status=0x0, info=-1}, 0x0, ) == 0x0
3982	NtDeviceIoControlFile (428, 0, 0x0, 0x0, 0x2346, "\6\0\0\0\0\0\0\0", 8, 0, ... {status=0x0, info=8}, 0x0, ) == 0x0

Table 6 – Strace output for WinDump program #3

Table 6 shows that the **NtCreateFile** function was used to create a file with an associated file handle of **428**. The file handle is a number that is assigned to every created or opened file. The file handle is used because it is easier to reference a file through the use of a number rather than the use of the filename. The **NtDeviceIoControlFile** function [30] was used multiple times to write the data that was captured from the network device to the file handle **428**.

No.	"WinDump.txt" Strace Output
3984	NtCreateFile (0x40100080, {24, 24, 0x42, 0, 1244428, "windump-capture"}, 0x0, 128, 3, 5, 96, 0, 0, ... 412, {status=0x0, info=3}, ) == 0x0

Table 7 – Strace output for WinDump program #4

Table 7 shows that the file "windump-capture" was created because the WinDump program was run with the "-w" option (show in Figure 67). The file handle **412** was assigned to this "windump-capture" file.

No.	"WinDump.txt" Strace Output
3986	NtWriteFile (412, 0, 0, 0, "\324\303\262\241\2\0\4\0\0\0\0\0\0\0\0\0\0\0\0\1\0\0\0", 24, 0x0, 0, ... {status=0x0, info=24}, ) == 0x0

Table 8 – Strace output for WinDump program #5

Table 8 shows that the **NtWriteFile** function was used to write the tcpdump file header (\324\303\262\241\2\0\4\0\0\0\0\0\0\0\0\0\0\0\0\1\0\0\0) to the file handle **412**. The file header is in octal format. Using the Windows XP calculator tool, the file header was translated to "D4 C3 B2 A1 02 00 04 00 00 00 00 00 00" in hexadecimal format.

Figure 72 shows the **khexedit** of the "windump-capture" file. The highlighted area displayed the tcpdump file header in hexadecimal format.

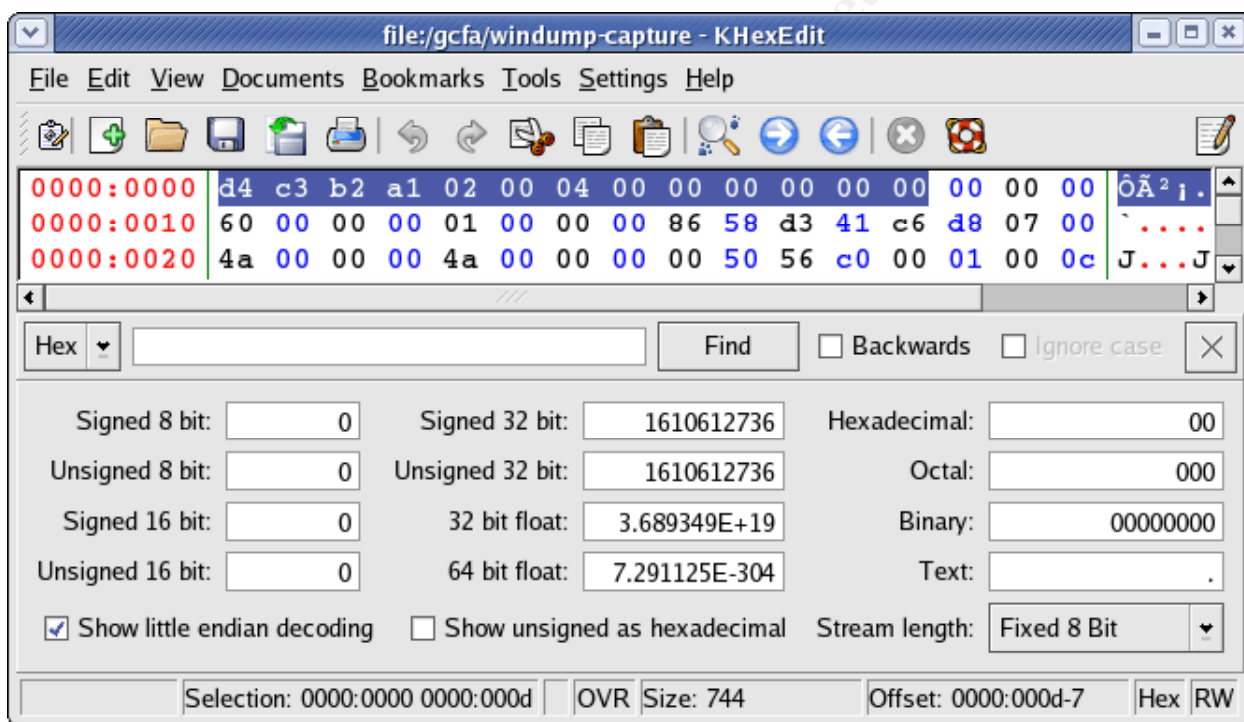


Figure 72 – Khexedit of the windump-capture file #1

No.	"WinDump.txt" Strace Output
3988	NtReadFile (428, 0, 0, 0, 256000, 0x0, 0, ... {status=0x0, info=0}, "", ) == 0x0
3989	NtWaitForSingleObject (424, 0, {-10000000, -1}, ... ) == 0x102
3990	NtReadFile (428, 0, 0, 0, 256000, 0x0, 0, ... {status=0x0, info=192}, "i\235\325A\270\314\3\0J\0\0\0J\0\0\0\24\0\0\0\0PV\300\0\10\0\14)\275\343\244\10\0E\0\0<\4\33\0\0\200\1\261J\300\250\2\12\300\250\2\1\10\0F\2\0\5\0abcdefghijklmnopqrstuvwabcdefghi\0\0i\235\325A\12\316\3\0J\0\0\0J\0\0\0\24\0\0\0\14)\275\343\244\0PV\300\0\10\0E\0\0<\361=\0\0@1\4(\300\250\2\1\300\250\2\12\0\0N\2\0\5\0abcdefghijklmnopqrstuvwabcdefghi\0\0", ) == 0x0



3991	NtWriteFile (412, 0, 0, 0, "i\235\325A\270\314\3\0J\0\0\0J\0\0\0", 16, 0x0, 0, ... {status=0x0, info=16}, ) == 0x0
3992	NtWriteFile (412, 0, 0, 0, "\0PV\300\0\10\0\14)\275\343\244\10\0E\0\0<\4\33\0\0\200\1\261J\300\250\2\12\300\250\2\1\10\0F\2\0\5\0abcdefghijklmnopqrstuvwabcdefghi", 74, 0x0, 0, ... {status=0x0, info=74}, ) == 0x0
3993	NtWriteFile (412, 0, 0, 0, "i\235\325A\12\316\3\0J\0\0\0J\0\0\0", 16, 0x0, 0, ... {status=0x0, info=16}, ) == 0x0
3994	NtWriteFile (412, 0, 0, 0, "\0\14)\275\343\244\0PV\300\0\10\10\0E\0\0<\361=\0\0@1\4(\300\250\2\1\300\250\2\12\0\0N\2\0\5\0abcdefghijklmnopqrstuvwabcdefghi", 74, 0x0, 0, ... {status=0x0, nfo=74}, ) == 0x0

Table 9 – Strace output for WinDump program #6

Table 9 shows that the **NtReadFile** function was used to read the data from the file handle **428** and then **NtWriteFile** function was used to write the data to the “windump-capture” file (file handle **412**). The same process was repeated again in Table 10, Table 11 and Table 12.

Figure 73 shows another **khxedit** of the “windump-capture” file. The highlighted area in this figure contained the same data that was written by the **NtWriteFile** function (No. 3992) in Table 8. This confirmed the write process.

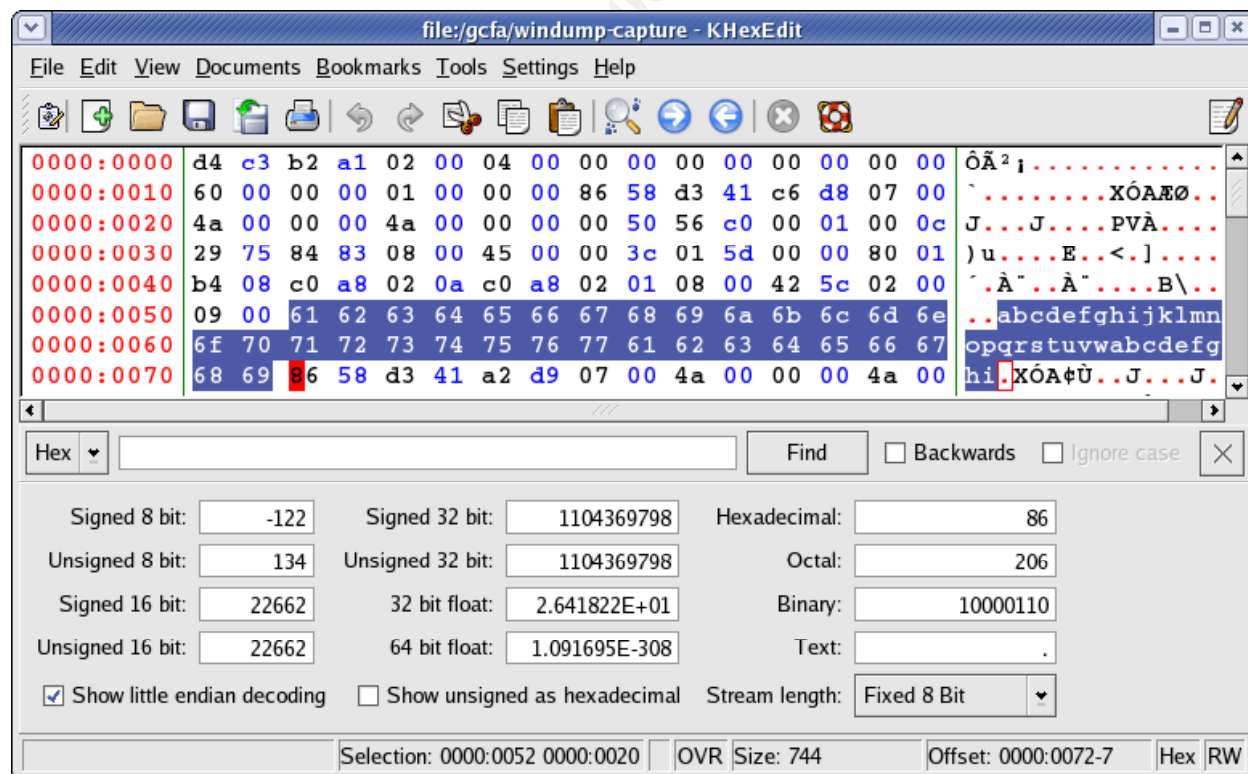


Figure 73 – Khxedit of the windump-capture file #2

No.	"WinDump.txt" Strace Output
-----	-----------------------------

3996	NtReadFile (428, 0, 0, 0, 256000, 0x0, 0, ... {status=0x0, info=192}, "j\235\325A\350\250\3\0J\0\0\0J\0\0\0\24\0\0\0\0PV\300\0\10\0\14)\275\343\244\10\0E\0\0<\4\34\0\0\200\1\261I\300\250\2\12\300\250\2\1\10\0E\2\0\6\0abcdefghijklmnopqrstuvwabcdefghi\0\0j\235\325A\211\251\3\0J\0\0\0J\0\0\0\24\0\0\0\0\14)\275\343\244\0PV\300\0\10\10\0E\0\0<\361>\0\0@1\4'\300\250\2\1\300\250\2\12\0\0M\2\0\6\0abcdefghijklmnopqrstuvwabcdefghi\0\0", ) == 0x0
3997	NtWriteFile (412, 0, 0, 0, "j\235\325A\350\250\3\0J\0\0\0J\0\0\0", 16, 0x0, 0, ... {status=0x0, info=16}, ) == 0x0
3998	NtWriteFile (412, 0, 0, 0, "\0PV\300\0\10\0\14)\275\343\244\10\0E\0\0<\4\34\0\0\200\1\261I\300\250\2\12\300\250\2\1\10\0E\2\0\6\0abcdefghijklmnopqrstuvwabcdefghi", 74, 0x0, 0, ... {status=0x0, info=74}, ) == 0x0
3999	NtWriteFile (412, 0, 0, 0, "j\235\325A\211\251\3\0J\0\0\0J\0\0\0", 16, 0x0, 0, ... {status=0x0, info=16}, ) == 0x0
4000	NtWriteFile (412, 0, 0, 0, "\0\14)\275\343\244\0PV\300\0\10\10\0E\0\0<\361>\0\0@1\4'\300\250\2\1\300\250\2\12\0\0M\2\0\6\0abcdefghijklmnopqrstuvwabcdefghi", 74, 0x0, 0, ... {status=0x0, info=74}, ) == 0x0

Table 10 – Strace output for WinDump program #7

No.	"WinDump.txt" Strace Output
4002	NtReadFile (428, 0, 0, 0, 256000, 0x0, 0, ... {status=0x0, info=192}, "k\235\325A\350\251\3\0J\0\0\0J\0\0\0\24\0\0\0\0PV\300\0\10\0\14)\275\343\244\10\0E\0\0<\4\35\0\0\200\1\261H\300\250\2\12\300\250\2\1\10\0D\2\0\7\0abcdefghijklmnopqrstuvwabcdefghi\0\0k\235\325A\200\252\3\0J\0\0\0J\0\0\0\24\0\0\0\0\14)\275\343\244\0PV\300\0\10\10\0E\0\0<\361?\0\0@1\4&\300\250\2\1\300\250\2\12\0\0L\2\0\7\0abcdefghijklmnopqrstuvwabcdefghi\0\0", ) == 0x0
4003	NtWriteFile (412, 0, 0, 0, "k\235\325A\350\251\3\0J\0\0\0J\0\0\0", 16, 0x0, 0, ... {status=0x0, info=16}, ) == 0x0
4004	NtWriteFile (412, 0, 0, 0, "\0PV\300\0\10\0\14)\275\343\244\10\0E\0\0<\4\35\0\0\200\1\261H\300\250\2\12\300\250\2\1\10\0D\2\0\7\0abcdefghijklmnopqrstuvwabcdefghi", 74, 0x0, 0, ... {status=0x0, info=74}, ) == 0x0
4005	NtWriteFile (412, 0, 0, 0, "k\235\325A\200\252\3\0J\0\0\0J\0\0\0", 16, 0x0, 0, ... {status=0x0, info=16}, ) == 0x0
4006	NtWriteFile (412, 0, 0, 0, "\0\14)\275\343\244\0PV\300\0\10\10\0E\0\0<\361?\0\0@1\4&\300\250\2\1\300\250\2\12\0\0L\2\0\7\0abcdefghijklmnopqrstuvwabcdefghi", 74, 0x0, 0, ... {status=0x0, info=74}, ) == 0x0

Table 11 – Strace output for WinDump program #8

No.	"WinDump.txt" Strace Output
-----	-----------------------------

4008	NtReadFile (428, 0, 0, 0, 256000, 0x0, 0, ... {status=0x0, info=192},  "1\235\325A'\251\3\0J\0\0\0J\0\0\0\24\0\0\0\0PV\300\0\10\0\14)\275\ 343\244\10\0E\0\0<\4\36\0\0\200\1\261G\300\250\2\12\300\250\2\1\10\ \0C\2\0\10\0abcdefghijklmnopqrstuvwabcdefghi\0\01\235\325A\314\251\ \3\0J\0\0\0J\0\0\0\24\0\0\0\0\14)\275\343\244\0PV\300\0\10\10\0E\0\ 0<\361@ \0\0@ \1\4%\300\250\2\1\300\250\2\12\0\0K\2\0\10\0abcdefghijklmnop jk lmnopqrstuvwabcdefghi\0\0", ) == 0x0
4009	NtWriteFile (412, 0, 0, 0, "1\235\325A'\251\3\0J\0\0\0J\0\0\0", 16, 0x0, 0, ... {status=0x0, info=16}, ) == 0x0
4010	NtWriteFile (412, 0, 0, 0, "\0PV\300\0\10\0\14)\275\343\244\10\0E\ 0\0<\4\36\0\0\200\1\261G\300\ 250\2\12\300\250\2\1\10\0C\2\0\10\0abcdefghijklmnopqrstuvwabcdef ghi", 74, 0x0, 0, ... {status=0x0, info=74}, ) == 0x0
4011	NtWriteFile (412, 0, 0, 0, "1\235\325A\314\251\3\0J\0\0\0J\0\0\0", 16, 0x0, 0, ... {status=0x0, info=16}, ) == 0x0
4012	NtWriteFile (412, 0, 0, 0, "\0\14)\275\343\244\0PV\300\0\10\10\0E\0\0<\361@ \0\0@ \1\4%\300\250\ \2\1\300\250\2\12\0\0K\2\0\10\0abcdefghijklmnopqrstuvwabcdefghi", 74, 0x0, 0, ... {status=0x0, info=74}, ) == 0x0

Table 12 – Strace output for WinDump program #9

© SANS Institute 2000

## Program Identification

The WinDump version 3.8.3 beta source code (WDumpSrc.zip) was downloaded from <http://windump.polito.it/install/default.htm>. The WinPcap version 3.1 beta 2 source code (3.1beta2-WpcapSrc.zip) was downloaded from <http://windump.polito.it/misc/bin/>. Both of these source codes were uncompressed to the C:\Windump folder on the Windows 2000 Workstation VMWare image. In order to compile the WinDump, the WinPcap source code must be in the same root folder as the WinDump source code [31].

The Microsoft Visual C++ 6.0 Professional Edition compiler (January 2001) was installed on the Windows 2000 Workstation VMWare image. This compiler was needed to compile the WinDump source code [31]. Furthermore, the WinDump readme.w32 file indicates that the November 2001 (or late) edition of the Microsoft Platform SDK is required. The Windows Server 2003 Platform SDK (February 2003) was downloaded from <http://www.microsoft.com/msdownload/platformsdk/sdkupdate/psdk-full.htm>. This SDK was installed to the C:\Program Files\Microsoft SDK directory on the Windows 2000 Workstation VMWare image. Once the SDK was installed, the Microsoft Visual C++ 6.0 must be configured to use the Microsoft Platform SDK. This configuration was done in the Microsoft Visual C++ 6.0 tools, options and directories setting.

The WinDump project file was loaded into Microsoft Visual C++ 6.0 compiler as indicated on the WinDump site [31]. The WinDump project file contains all the source codes that must be compiled in order to build the WinDump executable file. The compiler can compile a debug or a release version of the WinDump program:

- The debug version of any program is compiled with symbols and no optimization. This version of the program is used primarily for debugging purposes only [32]. The debug version is larger in size than the release version.
- The release version of any program is compiled without symbols and it is fully optimized [32]. The release version is the final release version of the program.

The Microsoft Visual C++ 6.0 was configured (under the Build option in the menu bar and Set Active Project Configuration option) to compile the release version WinDump source code since the objective was to determine if this compiled WinDump was the same as the one used by Mr. Lawrence.

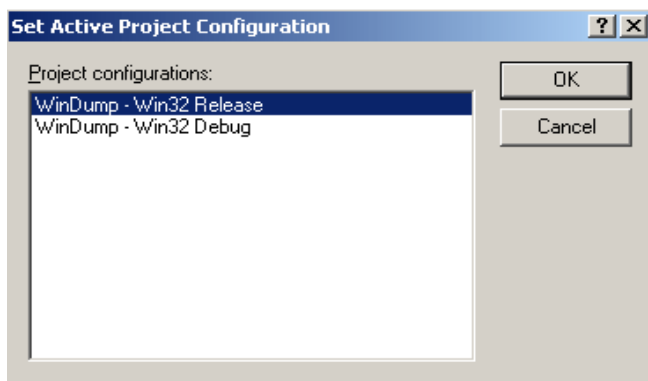


Figure 74 – Compile the release version of WinDump

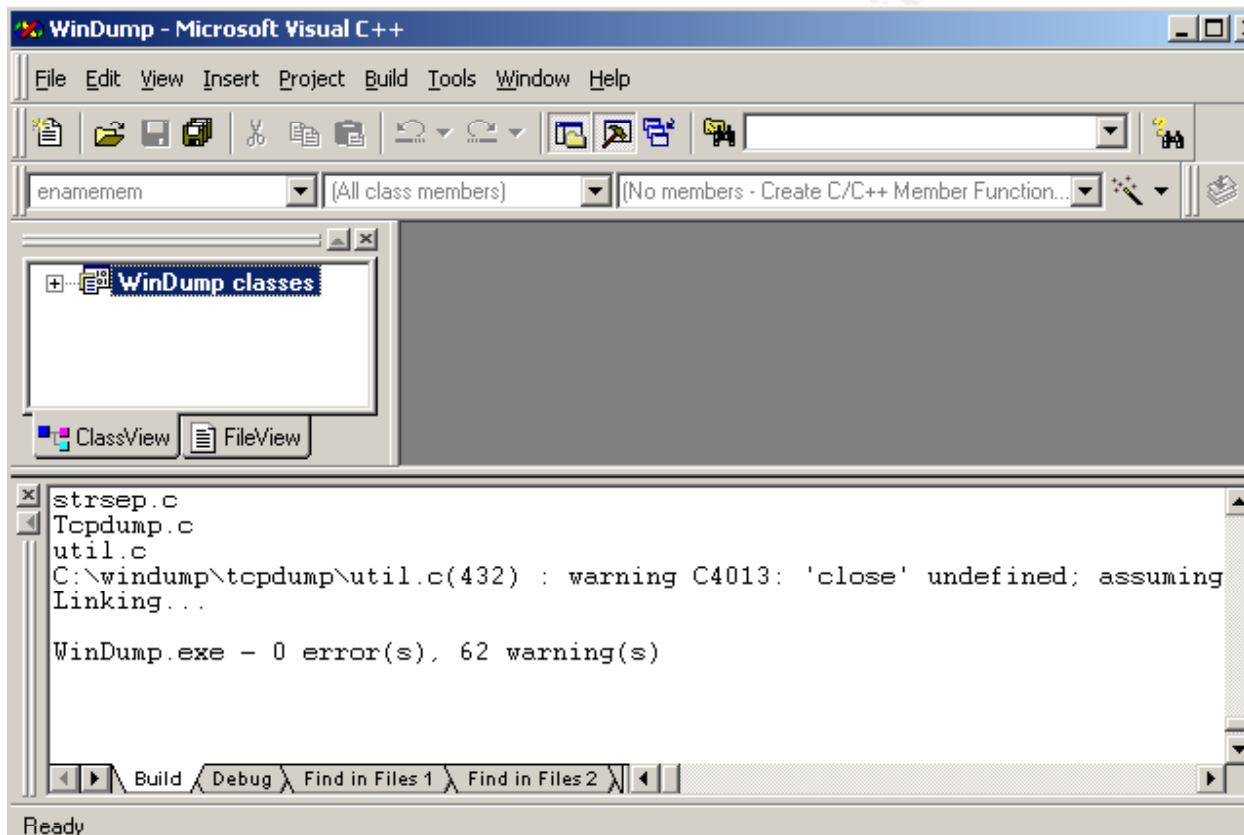
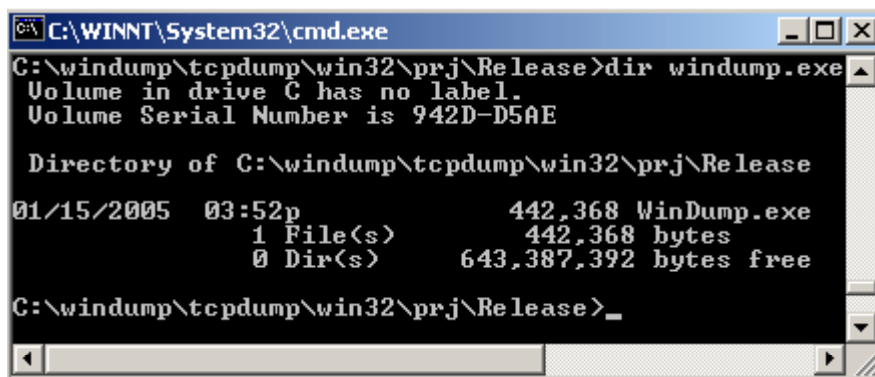


Figure 75 – Compiling WinDump (Release version)

Figure 75 shows the Microsoft Visual C++ 6.0 after compiling the WinDump. The WinDump.exe executable program was built in the C:\windump\tcpdump\win32\prj\Release folder as shown in Figure 76.

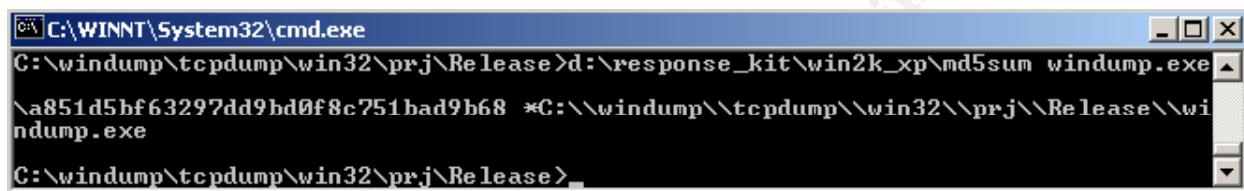


```
C:\WINNT\System32\cmd.exe
C:\windump\tcpdump\win32\prj\Release>dir windump.exe
Volume in drive C has no label.
Volume Serial Number is 942D-D5AE

Directory of C:\windump\tcpdump\win32\prj\Release
01/15/2005  03:52p                442,368 WinDump.exe
               1 File(s)                442,368 bytes
               0 Dir(s)        643,387,392 bytes free

C:\windump\tcpdump\win32\prj\Release>
```

Figure 76 – The compiled WinDump program (Release version)



```
C:\WINNT\System32\cmd.exe
C:\windump\tcpdump\win32\prj\Release>d:\response_kit\win2k_xp\md5sum windump.exe
a851d5bf63297dd9bd0f8c751bad9b68 *C:\windump\tcpdump\win32\prj\Release\windump.exe

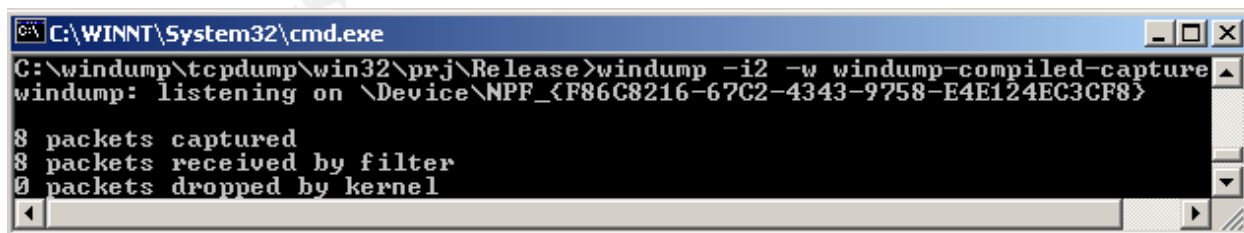
C:\windump\tcpdump\win32\prj\Release>
```

Figure 77 – MD5 hash of the compiled WinDump program (Release version)

Figure 76 shows that the compiled WinDump program was 442368 bytes in size. Figure 77 shows the MD5 hash of this compiled program. Both the MD5 hash and the size of the file were different from the WinDump program that was used by Mr. Lawrence. However, this does not indicate that the two WinDump programs were different. If the WinDump program used by Mr. Lawrence was compiled with a different version of the Microsoft Platform SDK, then different libraries would have been used and thus the size of the program would be different. Therefore, it was necessary to test the compiled WinDump program to determine if the output was the same as previous.

The compiled WinDump program was run with the same options as shown in Figure 67.

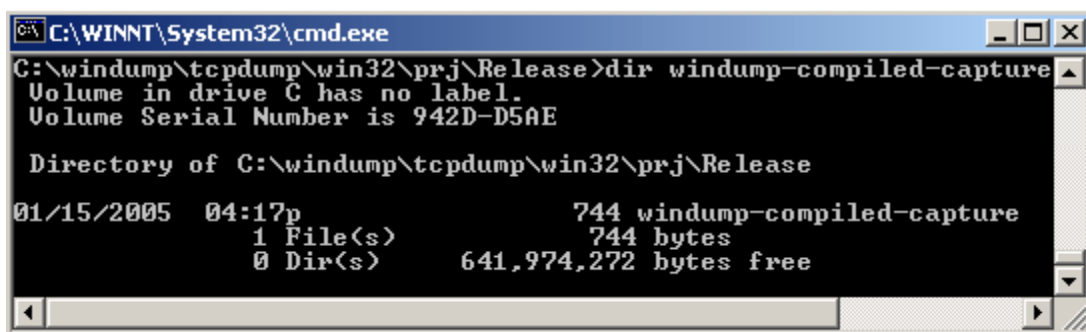
Again, a PING to 192.168.2.1 (the gateway) was initiated. This was done to generate network packets so that WinDump can capture these packets.



```
C:\WINNT\System32\cmd.exe
C:\windump\tcpdump\win32\prj\Release>windump -i2 -w windump-compiled-capture
windump: listening on \Device\NPF_{F86C8216-67C2-4343-9758-E4E124EC3CF8}

8 packets captured
8 packets received by filter
0 packets dropped by kernel
```

Figure 78 – Testing the compiled WinDump program (Release version)



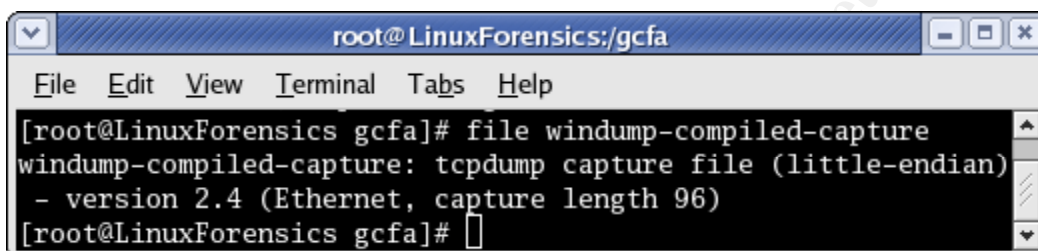
```
C:\WINNT\System32\cmd.exe
C:\windump\tcpdump\win32\prj\Release>dir windump-compiled-capture
Volume in drive C has no label.
Volume Serial Number is 942D-D5AE

Directory of C:\windump\tcpdump\win32\prj\Release

01/15/2005  04:17p                744 windump-compiled-capture
               1 File(s)                744 bytes
               0 Dir(s)          641,974,272 bytes free
```

Figure 79 – The WinDump capture file

Figure 79 shows the compiled WinDump's output file "windump-compiled-capture". This file was copied to the Linux Forensics machine.



```
root@LinuxForensics:/gcfa
File Edit View Terminal Tabs Help
[root@LinuxForensics gcfa]# file windump-compiled-capture
windump-compiled-capture: tcpdump capture file (little-endian)
- version 2.4 (Ethernet, capture length 96)
[root@LinuxForensics gcfa]#
```

Figure 80 – File type classification on the WinDump capture file

Figure 80 shows that this file was a tcpdump capture file. The **Ethereal** utility was used to further analyze this file.

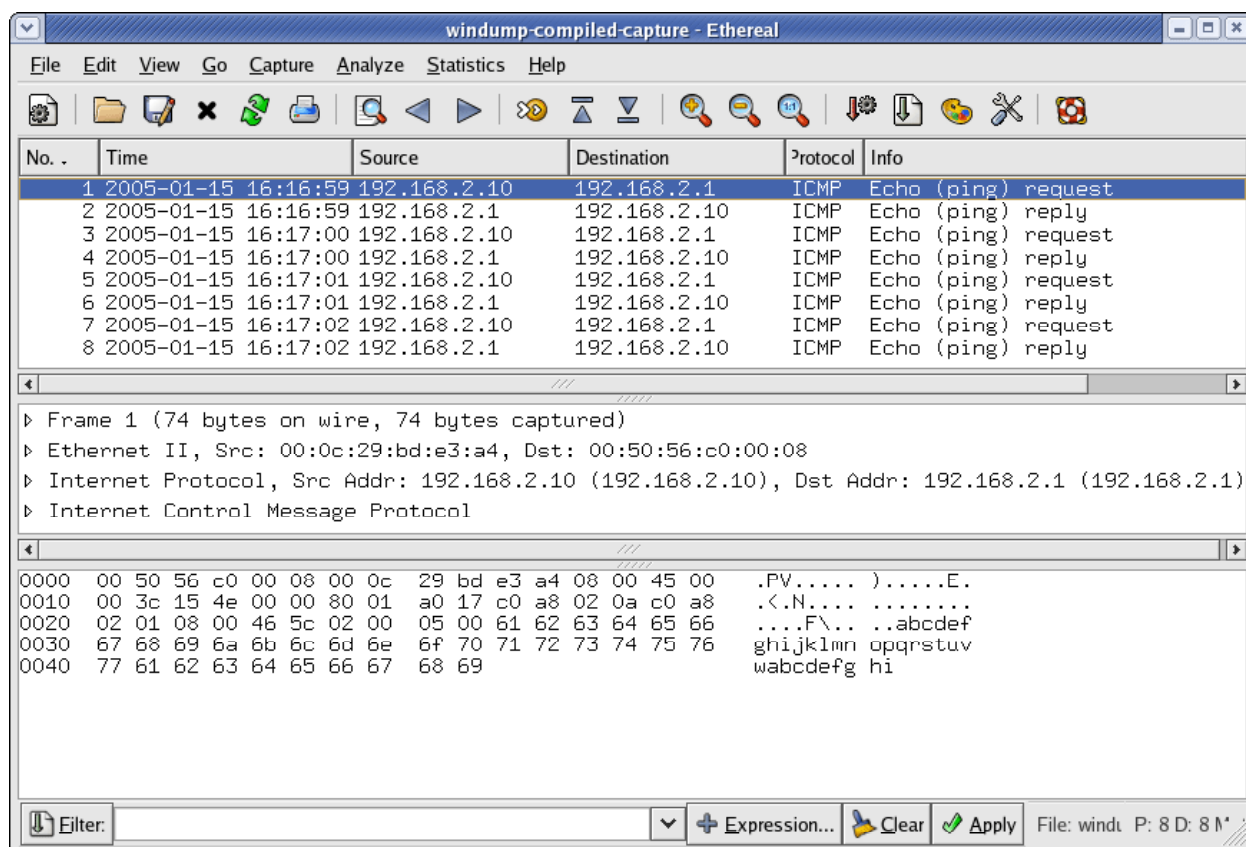


Figure 81 – Ethereal of the compiled WinDump capture file

Figure 81 shows that there were 4 ICMP Echo request packets and 4 ICMP Echo reply packets. This result was the same as the result produced by the WinDump program used by Mr. Lawrence. Therefore, it was concluded that the compiled WinDump program was the same as the program used by Mr. Lawrence

The source code for WinPcap\_3\_1\_beta\_3.exe could not be found. However, this program was easily identified because of the file name. The google search engine was used to locate the file name WinPcap\_3\_1\_beta\_3.exe. This file was downloaded from <http://www.oltenia.ro/download/pub/windows/network%20tools/ethereal/>. The downloaded file has the same file size as the one indicated by the timeline in Figure 62. It was determined in the examination details section that the downloaded file was the same file as the file indicated by the timeline. The strace output in the forensic details section indicated that this program installed certain libraries. It was also shown that the WinDump program used these libraries. The WinDump site <http://windump.polito.it/> indicates that the WinDump program requires the WinPcap libraries to function, thus the WinPcap\_3\_1\_beta\_3.exe file was identified.



## Legal Implications

It was determined that Mr. Lawrence, a sales representative at his firm, had used a wiretapping program to intercept an email transaction between Ms. Conlay and her friend. The purpose of his action was to determine personal information on Ms. Conlay for personal use. In this particular act, Mr. Lawrence has violated the Invasion of Privacy Law in Canada. Subsection 184.(1), Chapter C-46 of the Canada Criminal Code states that [33]:

184. (1) Every one who, by means of any electro-magnetic, acoustic, mechanical, or other device, wilfully intercepts a private communication is guilty of an indictable offence and liable to imprisonment for a term not exceeding five years.

It should be noted that Subsection 184 (1) does not apply if Ms. Conlay had given the consent to Mr. Lawrence to wiretap her private communication. Subsection 184.(2), Chapter C-46 states that [33]:

184.(2) Subsection (1) does not apply to

(a) a person who has the consent to intercept, express or implied, of the originator of the private communication of the person intended by the originator thereof to receive it;

It was obvious that Ms. Conlay would not have given the consent to Mr. Lawrence, as the email transaction was a private communication between Ms. Conlay and her friend.

It should also be noted that Subsection 184.(2) (a), Chapter C-46 of the Canada Criminal Code does not apply to system administrators who are intercepting or monitoring their network traffic for the purpose of protecting their network from being used illegally, as indicated by Subsection 184.(2) (e), Chapter C-46 of the Canada Criminal Code. However, this exception should be used with care, as it does not permit a system administrator to gain information on individuals for personal use. Since Mr. Lawrence was a sales representative at his company, he would not fall under this exception. Furthermore, Mr. Lawrence used the wiretapping program to obtain information on Ms. Conlay for personal use only.

Subsection 184.(2) (e), Chapter C-46 states that [33]:

184. (2) Subsection (1) does not apply to

(e) a person, or any person acting on their behalf, in possession or control of a computer system, as defined in Subsection 342.1(2), who intercepts a private communication originating from, directed to or transmitting through that computer system, if the interception is reasonably necessary for

(i) managing the quality of service of the computer system as it relates to performance factors such as the responsiveness and capacity of the system as well as the integrity and availability of the system and data, or

(ii) protecting the computer system against any act that would be an offence under subsection 342.1(1) or 430(1.1).

Subsection 342.1(1), Chapter C-46 states that [34]:

342.1(1) Every one who, fraudulently and without colour of right,

(a) obtains, directly or indirectly, any computer service,

(b) by means of an electro-magnetic, acoustic, mechanical or other device, intercepts

Subsection 342.1(2), Chapter C-46 defines “computer system” as [34]:

“computer system” means a device that, or a group of interconnected or related devices one or more of which,

(a) contains computer programs or other data, and

(b) pursuant to computer programs,

(i) performs logic and control, and

(ii) may perform any other function;

Besides violating these laws, it was also shown in the investigation that Mr. Lawrence had harassed Ms. Conlay. The documents on the USB JumpDrive clearly indicate Ms. Conlay’s intent of having no relationship with Mr. Lawrence. However, he did not receive Ms. Conlay’s intent in a welcome manner. He even indicated in one of the documents that there may have been a bad batch of coffee and sarcastically hoped that Ms. Conlay did drink not the coffee. This implied that he has the intent to harm Ms. Conlay. By doing this, Mr. Lawrence has violated Subsection 264.1(1), Chapter C-46 of the Canada Criminal Code, which states that [35];

264.1 (1) Every one commits an offence who, in any manner, knowingly utters, conveys or causes any person to receive a threat

(a) to cause death or bodily harm to any person;

(b) to burn, destroy or damage real or personal property; or

(c) to kill, poison or injure an animal or bird that is the property of any person.

(2) Every one who commits an offence under paragraph (1)(a) is guilty of

(a) an indictable offence and liable to imprisonment for a term not exceeding five years; or

(b) an offence punishable on summary conviction and liable to imprisonment for a term not exceeding eighteen months.

In addition to violating laws that can result in imprisonment, Mr. Lawrence may also have violated company policies. Most companies have acceptable use policies, which usually define the acceptable and unacceptable use of the computing and network resources in the company. Therefore, policies such as no unauthorized installation of software, no unauthorized use of software and no unauthorized wiretapping of the company's networks would usually be included in the acceptable use policy. Mr. Lawrence would have easily violated these policies by installing the WinPCap libraries and wiretapping the company's network using WinDump.

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

## Recommendations

The investigation is now complete. The following are some recommendations as to what should happen next.

It was shown in the investigation that Mr. Lawrence had installed and used a wiretapping device to intercept an email communication between Ms. Conlay and her friend. He did this without Ms. Conlay's or his company's consent.

The first recommendation is to contact the firewall administrator at CC Terminals for the firewall logs dating from October 24<sup>th</sup> 2004 to October 29<sup>th</sup> 2004. The purpose of this is to determine two things:

- If Mr. Lawrence had download the WinDump and WinPcap programs
- If Mr. Lawrence had visited the <http://www.mapblast.com>

If Mr. Lawrence had downloaded the WinDump and WinPcap programs at work, the firewall logs would indicate these activities. Furthermore, if Mr. Lawrence had visited the <http://www.mapblast.com> at work to determine Ms. Conlay's meeting location, the firewall logs would also indicate that activity. This information can be used to further substantiate Mr. Lawrence's actions.

The second recommendation is to conduct an interview with Mr. Lawrence, the security administrator, and a HR representative. The purpose of the interview is to obtain a confession from Mr. Lawrence, stating that he was responsible for the actions identified in this investigation. In this particular case, the approach of the interview process will be direct because there is substantial evidence to prove that Mr. Lawrence had done something unacceptable. The following indicates the interview questions for Mr. Lawrence and the reasons for the questions:

1. We found a USB JumpDrive in your cubicle. There were three Microsoft Word files on this USB Jumpdrive drive and the properties of these files indicate that you are the owner. Furthermore, these files contain the emails that you had sent to Ms. Conlay. Ms. Conlay also has copies of these emails in her mailbox. Therefore, can you confirm that this is your USB JumpDrive?
  - The purpose of this question is to establish the ownership of this USB JumpDrive. By mentioning the contents of the USB JumpDrive and the fact that Ms. Conlay probably has copies of the emails, it would be difficult for Mr. Lawrence to deny ownership. The expected answer from Mr. Lawrence would be "yes".
2. How did you find Ms. Conlay's private email address [flowergirl96@hotmail.com](mailto:flowergirl96@hotmail.com)? It is very unlikely you could have guessed it.
  - Mr Lawrence will likely have no answer to this question. Therefore, his likely

respond would be “I can’t remember or I don’t know”. The purpose of this question is to increase the anxiety of this interview process.

3. Do you know that you can undelete files that have been deleted? We were able to recover 4 deleted files from your USB JumpDrive. Since this is your USB JumpDrive, can you please explain what these files are used for?
  - At this point, Mr. Lawrence will likely claim the fact that he has no knowledge of these files. The purpose of this question is to show the ability of the investigation team who had investigated this incident. Mr. Lawrence should begin to feel some pressure.
4. Two of the four recovered files are WinPcap and WinDump. Do you know how to use WinPcap and WinDump?
  - Again, Mr. Lawrence will likely continue to deny the fact that he knows how to use these two programs. However, the purpose of this question is to disclose the identification of the two recovered executable programs.
5. The other recovered file is a WinDump network capture file. This WinDump network capture file contains an email conversation between Ms. Conlay and her friend. From this email, we were able to determine the meeting time and location and Ms. Conlay’s private email address. Is this how you discovered Ms. Conlay’s private email address? Is this how you were able to discover the meeting time and location? Would you like to add anything to this?
  - Without mentioning the last recovered file, there are enough evidences that have been presented to cause Mr. Lawrence to confess at this point of the interview process.

The questions below are likely not necessary. However, to complete the interview process, these questions should be asked.

6. The last recovered file is a map of the coffee shop. Did you find the meeting location through the use of the WinDump program and then use a web-based map finder to determine the location?
  - Again, there will have been enough evidence presented that have been presented. Mr. Lawrence will have difficulty denying his actions.
7. Do you know that it is a violation of the company acceptable use policy to use a network wiretapping program and that the consequence of violating such policy is termination of employment?
  - The purpose of this question is to re-iterate the consequence of violating the acceptable use policy. Mr. Lawrence would have already understood the

company's acceptable use policy and that it is in violation of the policy to use a network wiretapping program. If he did not know of the company's acceptable use policy, he would not have concealed his activities. He would not have deleted those files.

8. Do you know that according to the Canada Criminal Code, it is illegal to use a network wiretapping program and that the punishment can be imprisonment?
  9. Do you know that it is in violation of the company's harassment policy to harass another individual and that the consequence of this type of actions could be termination of employment? Do you know that according to the Canada Criminal Code, it is illegal to harass another individual and that the punishment can be imprisonment?
- The purpose of the questions #8 and #9 is to inform Mr. Lawrence that according to the Canada Criminal Code, it is illegal to use a network wiretapping program and the potential punishment is imprisonment.

It is in the interest of CC Terminal to terminate Mr. Lawrence as an employee especially given the fact that he may face a lawsuit from Ms. Conlay. It is within the rights of Ms. Conlay to inform law enforcement of this harassment. The law enforcement will investigate this incident and will likely require the cooperation of CC Terminal. Thus, all the evidences collected throughout this investigation will have to be turned over to the law enforcement using the proper chain of custody process.

© SANS Institute 2000-2005

## Additional Information

Stevens, Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, 1994. This book details the TCP/IP protocol suite. It present information the TCP, UDP, IP packets are constructed. It is a requirement to understand the TCP/IP protocols in order to analyze tcpdump network captures.

The <http://undocumented.ntinternals.net> website provides information on most of the Windows NT/2000 kernel functions. This information is useful during the analysis of a strace output. The strace utility, when executed in a Windows NT/2000 environment, will generate a strace output. The strace output contains many Windows NT/2000 kernel functions.

The <http://msdn.microsoft.com/library/> website provides additional information on the Windows NT/2000 kernel functions.

The <http://www.liutilities.com/products/wintaskspro/dlllibrary/> website provides information on some of the common Windows dynamic link libraries. A program, during execution, may call other Windows dynamic link libraries. This process is often seen in the strace output. Therefore, this site provides some definition of these dynamic link libraries.

The <http://laws.justice.gc.ca/en/C-46/index.html> website details the Canada Criminal Code. The information provided at this site is useful as it states what is legal and what is illegal in the context of invasion of privacy.



## List of References

- [1] SANS Institute, Track 8 – System Forensics, Investigation, and Response. Volume 8.1. SANS Press, June 29, 2004 (pg19)
- [2] “Detailed Explanation of FAT Boot Sector” Microsoft Corporation Help and Support Library. December 6, 2003 <<http://support.microsoft.com/kb/q140418/>>.
- [3] “Unicode Character Table: Basic Latin” J.R. Graphics. January, 2005 <[http://jrgraphix.net/research/unicode\\_blocks.php](http://jrgraphix.net/research/unicode_blocks.php)>.
- [4] “WinPcap Description” WinPcap: the Free Packet Capture Library for Windows November 04, 2004 <<http://winpcap.polito.it/>>.
- [5] “WinDump 3.8.3 beta download” WinDump: tcpdump for Windows May 15, 2004 <<http://windump.polito.it/default.htm>>.
- [6] “NtCreateFile” Microsoft Corporation MSDN Library. December 2004 <<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devnotes/winprog/ntcreatefile.asp>>.
- [7] “ZwCreateFile” Microsoft Corporation MSDN Library. November 23, 2004 <[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/kmarch/hh/kmarch/k111\\_80b1882a-8617-45d4-a783-dbc3bfc9aad4.xml.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/kmarch/hh/kmarch/k111_80b1882a-8617-45d4-a783-dbc3bfc9aad4.xml.asp)>.
- [8] “ZwSetInformationFile” Microsoft Corporation MSDN Library. November 23, 2004 <[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/kmarch/hh/kmarch/k111\\_91ac021a-37b3-4d2d-9369-c80659e0dcd7.xml.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/kmarch/hh/kmarch/k111_91ac021a-37b3-4d2d-9369-c80659e0dcd7.xml.asp)>.
- [9] “NtOpenFile” Microsoft Corporation MSDN Library. December 2004 <<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devnotes/winprog/ntopenfile.asp>>.
- [10] “ZwQueryInformationFile” Microsoft Corporation. MSDN Library. November 23, 2004 <[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/kmarch/hh/kmarch/k111\\_822ab812-a644-4574-8d89-c4ebf5b17ea5.xml.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/kmarch/hh/kmarch/k111_822ab812-a644-4574-8d89-c4ebf5b17ea5.xml.asp)>.
- [11] Nowak, Tomasz. Undocumented Functions for Microsoft Windows NT/2000. January 9 2001 <<http://undocumented.ntinternals.net/>>.
- [12] Shreshtha Takshak, and Pradeep Gururani. “Dissecting IP Helper APIs” DeveloperIQ. November 11, 2004 <[http://www.developeriq.com/articles/view\\_article.php?id=151](http://www.developeriq.com/articles/view_article.php?id=151)>.
- [13] “INFO: Implementing Internet Pings Using Icmp.dll” Microsoft Corporation Help and

Support Library. February 9, 2000 <<http://support.microsoft.com/kb/q170591/>>.

[14] “mprapi – mprapi.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/mprapi/>>.

[15] “samlib – samlib.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/samlib/>>.

[16] “netapi32 – netapi32.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/netapi32/>>.

[17] “secur32 – secur32.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/secur32/>>.

[18] “netrap – netrap.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/netrap/>>.

[19] “dnsapi – dnsapi.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/dnsapi/>>.

[20] “activeds – activeds.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/activeds/>>.

[21] “Determining the version of ADSI that is installed on your computer” Microsoft Corporation Help and Support Library. July 13, 2004  
<<http://support.microsoft.com/kb/216290/EN-US/>>.

[22] “rtutils – rtutils.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/rtutils/>>.

[23] “setupapi – setupapi.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/setupapi/>>.

[24] “userenv – userenv.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/userenv/>>.

[25] “rasapi32 – rasapi32.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/rasapi32/>>.

[26] “rasman – rasman.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/rasman/>>.

[27] “tapi32 – tapi32.dll – DLL Information” Uniblue WinTasks DLL Library. 2005  
<<http://www.liutilities.com/products/wintaskspro/dlllibrary/tapi32/>>.

[28] “dhcpcsvc – dhcpcsvc.dll – DLL Information” Uniblue WinTasks DLL Library. 2005

<<http://www.liutilities.com/products/wintaskspro/dlllibrary/dhpcsvc/>>.

[29] "download clbcatq.dll file information" E-systems Download Center. 2005  
<<http://www.e-systems.ro/download-dll/clbcatq.dll/>>.

[30] "NtDeviceIoControlFile" Ntinternals.net Undocumented functions of NTDLL.  
February 11, 2001  
<<http://undocumented.ntinternals.net/UserMode/Undocumented%20Functions/NT%20Objects/File/NtDeviceIoControlFile.html>>.

[31] "Compiling WinDump" WinDump: tcpdump for Windows. March 14, 2002  
<<http://windump.polito.it/docs/compile.htm>>.

[32] "Debug and Release Configurations" Microsoft Corporation MSDN Library. 2005  
<[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsdebug/html/\\_core\\_Debug\\_Build\\_Versus\\_Release\\_Build.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsdebug/html/_core_Debug_Build_Versus_Release_Build.asp)>.

[33] "Interception of Communications, Chapter 46, Section 184". Canada Criminal Code <<http://laws.justice.gc.ca/en/C-46/42172.html>>.

[34] "Unauthorized use of computer, Chapter 46, Section 342.1". Canada Criminal Code  
<<http://laws.justice.gc.ca/en/C-46/42972.html>>.

[35] "Uttering Threats, Chapter 46, Section 264.1 (1)". Canada Criminal Code  
<<http://laws.justice.gc.ca/en/C-46/42515.html>> .

© SANS Institute 2000 - 2005