



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>

Computer forensics investigation

Image file analysis

GCFA Practical Version 2.0

Michael Spellane

February 27, 2005

Abstract

This practical assignment is version 2.0, the first step in attaining the highly regarded GCFA certification. I intend to demonstrate my ability to perform computer forensics, as well as my understanding of the laws covered pertaining to computer crimes. It is my intention to provide a clear view as to steps taken, tools that have been utilized, and laws that pertain to this investigation.

Executive Summary

I have completed an in depth investigation of evidence obtained from Mr. Robert Lawrence's desk, an image of a USB jump drive. Following this summary, you will find a detailed analysis regarding this investigation. Steps have been taken to maintain the integrity of the evidence, using all known methods to protect the image, and limit the evidence to as few people as possible. My analysis of the evidence has returned strong evidence that Mr. Lawrence has committed a computer crime. I have extracted evidence that Mr. Lawrence has "tapped" Ms. Conolay's computer system, monitoring her computer network traffic. He was able to view Ms. Conolay's personal email, violating her privacy. To accomplish this, he installed a computer program on Ms. Conolay's computer. The programs installed were:

- 1) Winpcap – A packet capturing library required by network sniffing programs such as TCPDUMP/WINDUMP, and ethereal, a graphical network protocol analyzer.
- 2) Windump – A packet capture program for Microsoft Windows platform computers. This was the program executed on Ms. Conolay's computer to capture all of her network traffic.

Upon capturing Ms. Conolay's traffic, Mr. Lawrence was able to read Ms. Conolay's personal email, which is held at hotmail.com. He was able to see where Ms. Conolay was going after work, what time she would be arriving, and with whom she was meeting. I have also recovered a map file, which displayed a graphical picture of the address to where Ms. Conolay was going to meet her friend for coffee.

Now, Microsoft Word documents were recovered from this image, and they contain messages to Ms. Conolay from Mr. Lawrence, indicating that he finds her attractive, and would like to enter into a personal relationship with her. Ms. Conolay rejected Mr. Lawrence's offers, but Mr. Lawrence was not pleased with this response. In the last Word Document, he has typed a message to Ms. Conolay that can be considered threatening in nature.

Mr. Lawrence deleted the files he used to capture the traffic, including the file that had the traffic saved in it, but through the use of computer forensics, and careful analysis, these files were able to be recovered and analyzed to see what actions were being performed by Mr. Lawrence.

It is my professional opinion that Mr. Lawrence has broken laws pertaining to computer crimes. These laws include:

Federal Wiretap Act – Violated when Mr. Lawrence installed the wiretap software, winpcap and utilized windump.

Electronic Communications Privacy Act – Violated when Mr. Lawrence read Ms. Conolay's private email, which is stored at hotmail.com.

Examination Details

The first step in this investigative journey is to obtain the evidence in question. The GCFA practical version 2 option I will be performing is to analyze an image file provided by the SANS institute. As such, I have obtained the image from <https://www.giac.org/GCFAPractical2.0-USBImageAndInfo.zip.gz> for further investigation. Here is the information about the evidence provided by the GCFA ver 2.0 practical:

Tag #: USBFD-64531026-RL-001

Description: 64M Lexar Media JumpDrive

Serial #: JDSP064-04-5000C

Image: USBFD-64531026-RL-001.img

MD5: 338ecf17b7fc85bbb2d5ae2bbc729dd5

Now that we have the image in hand, we can decide what needs to be accomplished with this file. The file was downloaded and saved to a directory on a Microsoft Windows machine. Next, I then mounted the SMB share from my Linux forensics workstation, which was loaded as a forensics workstation using the SANS track 8 CDROM provided to students attending the track 8 course:

Mount -t smbfs -o username=username(sanitized for the sake of my network) //my.net.1.13/c\$ /mnt/images

Here, we are mounting the share located on the Windows machine using the -t switch, which specifies the file system type, which, in this case, is SMBFS. This gives us the ability to view and interpret the server message block protocol, and SMB file system. The -o switch lets us specify a username to pass to the machine for authentication. We also provide the IP to the machine that we are connecting to, along with the share name. Let's verify the file type, and move on from there:

[root@LinuxForensics images]# file GCFAPractical2.0-USBImageAndInfo.zip.gz

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

As you can see from the output of the FILE command (explained further below), the file that was obtained from the SANS website is a GZIP compressed file. This file contains the file "USBFD-64531026-RL-001.img" within it. Since this file was downloaded onto a Windows machine, I have used WinZip to extract image file from compressed file. WinZip is a well known file compression/decompression utility used to compress or decompress files. This is useful when file sizes are too large for an individuals needs. WinZip can be downloaded for trial from <http://www.winzip.com> .

Upon extracting the image file, I have copied the "USBFD-64531026-RL-001.img" image file to my Linux Fedora Core forensic workstation. Now, I must find out the true type of file that was extracted. Once again, the FILE command was used to accomplish this task:

[root@LinuxForensics images]# file USBFD-64531026-RL-001.img
USBFD-64531026-RL-001.img: x86 boot sector

The file command works by testing "arguments" within a file, and will then classify the file as whichever file type the file command sees fit. We see from the output of the file command that the image file contains an x86 boot sector. The boot sector of a computer is a primary starting point for an OS. The operating system will start at the boot loader, and the machine will read the first 512 bytes of the disk, which is known as the boot sector. The first 512 Bytes (boot sector) will be loaded into memory and will then be executed. This will initiate the boot process. Next, we will want to verify the integrity of the image. This can be accomplished by viewing the MD5 hash of the file, and comparing it to the MD5 hash provided by SANS:

[root@LinuxForensics images]# md5sum USBFD-64531026-RL-001.img
338ecf17b7fc85bbb2d5ae2bbc729dd5 USBFD-64531026-RL-001.img

An md5 hash is a crypto logic value assigned to a given file to ensure the integrity of the file. The md5 hash algorithm produces a 128 bit "fingerprint" of a file, also known as a message digest. This ensures non-repudiation integrity of the file. To view the md5 hash value assigned to a given file, the md5sum utility can be used. md5sum is a utility that will check the md5 hash of a file. An md5 hash is comprised of 128 bit checksums. The md5sum output resulted in a hash value of **338ecf17b7fc85bbb2d5ae2bbc729dd5**. When we reference the hash value supplied by the SANS institute above, we see that the hash values match. **338ecf17b7fc85bbb2d5ae2bbc729dd5**

Next, we want to extract the logical partitions from the image file, but first we need to list all of the partitions within the image file to see exactly what we want to focus on. To accomplish this, we utilize the mmls utility.

```
[root@LinuxForensics images]# mmls -t dos USBFD-64531026-RL-001.img
```

DOS Partition Table

Units are in 512-byte sectors

Slot	Start	End	Length	Description
00:	-----	00000000000	00000000000	00000000001 Primary Table (#0)
01:	-----	00000000001	00000000031	00000000031 Unallocated
02:	00:00	00000000032	0000121950	0000121919 DOS FAT16 (0x04)

MMLS is a forensics utility that query's an image file, and prints the partition tables and disk labels. This command is very useful when attempting to determine at which sector a partition begins and ends. We see that there is a FAT16 file system on this image. This FAT16 partition appears to contain the largest amount of data on the image. We use the `-t dos` switch to tell mmls to utilize a dos based architecture for the file system. These partitions are used by ALL x86 based systems, including Linux. The FAT 16 file system was used on older computers, DOS, early windows, and for small partitions on modern systems, such as USB memory storage key's, or jumpdrives as they are often referred to. Remember, the evidence in which the image was taken was that of a USB jumpdrive. FAT 16 has a 2GB size limit, which makes it technically unfeasible to use on modern systems, where file systems are required to have larger sizes. FAT 16 uses a 16-bit binary number to hold the cluster numbers (assigned to a file). FAT links these cluster numbers to their respective files. FAT 16 volumes can hold a maximum of 65,526 clusters (2 to the 16th power). Moving on from our FAT16 background lesson, let's extract this partition from the image file using the dd command:

```
[root@LinuxForensics GCFA]# dd if=USBFD-64531026-RL-001.img bs=512  
skip=32 count=121919 of=sanswork
```

121919+0 records in

121919+0 records out

The dd command will copy a specified input file, the image file in hand, and output the contents to a file specified by the user. You may specify block sizes, starting sectors, and the length, in sectors, that will be copied to another file. This

will essentially copy, or “carve”, the partition that we have identified of as interest to an output file which can then be mounted, or can be used to run forensics tools against. A very attractive tool to run against these image files would be Autopsy, the forensic browser. This program provides a sexy graphical user interface for the command line tools found in the popular sleuth kit. This is a very important step in forensics analysis. Now, we have just carved out the FAT16 partition from the image file. You may have noticed that I have passed certain switches to the command; these switches instruct dd to perform the following:

“IF” is the input file name. This switch will be followed by a specified image file, and, as we know, the file that we are working with is USBFD-64531026-RL-001.img image file. This is the filename that we have passed to the command as its input file.

“BS” instructs DD to use a block size of 512 bytes. This option supercedes the switches IBS (input block size) and OBS (output block size) and assigns this value to BOTH the IBS and OBS.

“SKIP” tells DD to skip a given amount of sectors. Here, we have passed the numerical value of 32 to the command, instructing dd to skip the first 32 sectors of the image file. We have arrived at the number 32 by looking at the mmls output, where the starting sector of the FAT16 partition is 32. Now, why would we instruct dd to skip the first 32 sectors, if the sector we want to start at is labeled 32? Well, simple; much like in the HEX world, counting starts at 0, so, the numerical value of 32 is actually sector 31, if counting from 1. Ok, now we have an image of a FAT partition; we will mount this image to see what it contains to the naked eye. For this, I will use the mount command with the “read only” option, as not to cause any changes to the image, and maintain its integrity.

```
[root@LinuxForensics images]# mount -t auto -o  
loop,ro,noatime,noexec,nosuid  
sanswork /mnt/sans/
```

A quick ls -al of the mounted partition shows us what appears to be three Microsoft word documents, along with their respective file sizes. Running the FILE command against these files indicate that they are Microsoft Word documents, so, no surprises there.

```
[root@LinuxForensics root]# ls -al /mnt/sans/  
total 80  
drwxr-xr-x 2 root root 16384 Dec 31 1969 .  
drwxr-xr-x 6 root root 4096 Jan 15 10:33 ..  
-rwxr-xr-x 1 root root 19968 Oct 28 20:24 coffee.doc  
-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
```

Now, the UNIX `ls -al` command will print all files in the current directory as well as a long listing of these files and any subdirectories contained therein. Now, we have these three files that we have verified as being Microsoft word documents, residing in this partition. What should we do with these files next? Let's open up our favorite hex editor, KHEXEDIT, and examine the raw contents of her.doc. We have chosen to start with HER.DOC due to the fact that it seems to have been created first, as indicated by the timestamp Oct 25 @09:32. I have utilized the Linux hex editor KHEXEDIT to view this file, as well as further Word document files. The Linux KHEXEDIT program will allow us to view the raw data of binary files, and display HEX representations as well as plain text. This program comes in very handy when performing forensic analysis against Microsoft Word documents. This allows us to view the metadata layer of the file, the information that the file passes on to its founding program, Microsoft Word. Metadata can be used to alter or hide information, and should be carefully examined. The information contained at the metadata layer is not printed on the document itself. The metadata of a Microsoft word file contains sensitive information about the user and the user's resources. Some of this information includes the user's name, the users computer name, any file server names that the file may have been saved to, names of other document authors, OLE objects, ect. Remember, metadata is essentially data about data. An excellent write up about metadata can be found at:

<http://www.hsskgroup.com/attachments/articles/57/Metadata%2010%2020%2004.pdf>

We can see the entire message using KHEXEDIT, with the TEXT view. We will also be able to see the "branding" that Microsoft Word does on documents. This will tell us the path to which the file was saved, and can give us an indication of the owner of the document. This text would be visible under any word processor that could read Microsoft word documents.

**Hey I saw you the other day. I tried to say "hi", but you disappeared???
That was a nice blue dress you were wearing. I heard that your car was
giving you some trouble. Maybe I can give you a ride to work sometime, or
maybe we can get dinner sometime?**

Have a nice day

Ok, seems harmless enough, no plans at world domination or anything like that. Now, here is what the hex editor has displayed for us. This is a representation of the raw data, as well as the plain text. Let's see what else our hex editor will display for us:

```
0000:1400 .....@..`ňÿ..@.....|.....N.o.r.m.a.l.....  
0000:1440 Cj._H..aJ..mH..sH..tH.....D.A@òÿ|.D.....  
0000:1480 D.e.f.a.u.i.l.t..P.a.r.a.g.r.a.p.h..F.o.n.t.....R.i@óÿ_.R.....  
0000:14c0 .....T.a.b.l.e..N.o.r.m.a.l.....ö...4Ö.....l.4Ö.....aö.....
```



```

0000:1500 ....(k@ôÿÁ.(.....N.o. .L.i.s.t.....
0000:1540 ..ÿÿÿÿ..... ÿÿ.. z.....
0000:1580 .0.....0.....0.....
0000:15c0 .....0.....
0000:1600 M9.0.....M9.0.....
0000:1640 .....M9.0.....Ö.....
0000:1680 ....._8....._.....@..ñ
0000:16c0 ....ÿÿ....ÿ.....÷....._....._0....._(.....
0000:1700 _....._....._B....._.....S..
0000:1740 _.....¿.....Ë.....ÿ.....?....._.....
0000:1780 .....ÿÿ.....R.o.b.e.r.t. .L.a.w.r.e.n.c.e..
0000:17c0 .....å.....|.ÿ@.....lÛ.....
0000:1800 .....P....@..ÿÿ.....U.n.k.n.o.w.n.ÿÿ.....ÿÿ
0000:1840 .....ÿÿ....ÿÿ....ÿÿ....ÿÿ.....G.....Z. ....
0000:1880 ..ÿ.....T.i.m.e.s. .N.e.w. .R.o.m.a.n..5.....
0000:18c0 .....S.y.m.b.o.l...3&.....z. ....
0000:1900 ÿ.....A.r.i.a.l...?5.....z. ....ÿ.....C.o.
0000:1940 u.r.i.e.r. .N.e.w..."...1...._...h.....Ê.& Ê.&.....(...ê.
0000:1980 .....(...ê.....!....._.....
0000:19c0 .....
0000:1a00 .....
0000:1a40 .....
0000:1a80 .....
0000:1ac0 ..... ' ' ...r4....d.....
0000:1b00 .....
0000:1b40 .....3..._.....H.....)_ÿ....?
0000:1b80 ..ä...ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ. |.ÿÿ.....H.e.y. .l. .s.
0000:1bc0 a.w. .y.o.u. .t.h.e. .o.t.h.e.r. .d.a.y.....R.o.b.e.r.t. .L.
0000:1c00 a.w.r.e.n.c.e...R.o.b.e.r.t. .L.a.w.r.e.n.c.e.....

```

Ok, so, this document appears to be owned by one Robert Lawrence, as we can see from the last entry of the raw data. However, this metadata may be easily manipulated using various methods. I have tested this using Winhex on a windows machine by manipulating the section of metadata that contained my user name. Also, if you simply click File > Properties while inside the word document, this can be changed here as well. This data lies at offset 0000:1ac0 of the document.

Now we move on and perform the same steps on the other two document files. When displayed in a hex editor, the other two documents displayed the same metadata information entry of "Robert Lawrence", so we can infer that all of these documents are owned by this individual. Here is the text of the remaining two files, HEY.DOC and COFFEE.DOC respectively:

Hey! Why are you being so mean? I was just offering to help you out with your car! Don't tell me to get lost! You should give me a chance. I'm a

nice guy just trying to help you out, just because I think you're cute doesn't mean I'm weird. Perhaps coffee would be better, when would be a good time for you?

Ok, from Mr. Lawrence's response, it is clear that Ms. Conlay has refused Mr. Lawrence's offer, but Mr. Lawrence is remaining persistent.

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

Clearly Mr. Lawrence is agitated by Ms. Conlay's refusal to enter into a personal relationship with him. His response could be taken as a threat; he seems to be indicating that he will put something into Ms. Conlay's coffee that would have dissatisfactory results on her health.

Now we will create a timeline for this image using the ils command. ils will list the inode information of removed files (by default), and allow us to run the mactime utility against the outputted file. ils is a command that is included in the sleuth kit, a computer forensics suite of utilities.

[root@LinuxForensics images]# ils -f fat -m sanswork > sanswork.ils

Next, I ran mactime against the sanswork.ils file:

[root@LinuxForensics images]# mactime -b sanswork.ils -h -d

The Sleuth Kit mactime Timeline
Input Source: sanswork.ils

Date	Size	Type	Mode	UID	GID	Meta	File Name
Wed Oct 27 2004 00:00:00	0	.a.	-rwxrwxrwx	0	0	12	<sanswork-_INDUMP.EXE-dead-12>
Wed Oct 27 2004 00:00:00	0	.a.	-rwxrwxrwx	0	0	7	<sanswork-_INPCA~1.EXE-dead-7>
Wed Oct 27 2004 16:23:50	485810	.m..	-rwxrwxrwx	0	0	10	<sanswork-_INPCA~1.EXE-dead-10>
Wed Oct 27 2004 16:23:54	485810	..c.	-rwxrwxrwx	0	0	10	<sanswork-_INPCA~1.EXE-dead-10>
Wed Oct 27 2004 16:23:54	0	..c.	-rwxrwxrwx	0	0	7	<sanswork-_INPCA~1.EXE-dead-7>
Wed Oct 27 2004 16:23:56	0	.m..	-rwxrwxrwx	0	0	7	<sanswork-_INPCA~1.EXE-dead-7>
Wed Oct 27 2004 16:24:02	450560	.m..	-rwxrwxrwx	0	0	14	<sanswork-_INDUMP.EXE-dead-14>
Wed Oct 27 2004 16:24:04	450560	..c.	-rwxrwxrwx	0	0	14	<sanswork-_INDUMP.EXE-dead-14>
Wed Oct 27 2004 16:24:04	0	..c.	-rwxrwxrwx	0	0	12	<sanswork-_INDUMP.EXE-dead-12>
Wed Oct 27 2004 16:24:06	0	.m..	-rwxrwxrwx	0	0	12	<sanswork-_INDUMP.EXE-dead-12>
Thu Oct 28 2004 00:00:00	53056	.a.	-rwxrwxrwx	0	0	15	<sanswork-_apture-dead-15>
Thu Oct 28 2004 00:00:00	8814	.a.	-rwxrwxrwx	0	0	17	<sanswork-_ap.gif-dead-17>
Thu Oct 28 2004 00:00:00	485810	.a.	-rwxrwxrwx	0	0	10	<sanswork-_INPCA~1.EXE-dead-10>
Thu Oct 28 2004 00:00:00	450560	.a.	-rwxrwxrwx	0	0	14	<sanswork-_INDUMP.EXE-dead-14>
Thu Oct 28 2004 00:00:00	0	.a.	-rwxrwxrwx	0	0	16	<sanswork-_ap.gif-dead-16>

```
Thu Oct 28 2004 11:08:24,53056,..c,-rwxrwxrwx,0,0,15,<sanswork-_apture-dead-15>
Thu Oct 28 2004 11:11:00,53056,m..,-rwxrwxrwx,0,0,15,<sanswork-_apture-dead-15>
Thu Oct 28 2004 11:17:44,0,..c,-rwxrwxrwx,0,0,16,<sanswork-_ap.gif-dead-16>
Thu Oct 28 2004 11:17:44,8814,..c,-rwxrwxrwx,0,0,17,<sanswork-_ap.gif-dead-17>
Thu Oct 28 2004 11:17:46,8814,m..,-rwxrwxrwx,0,0,17,<sanswork-_ap.gif-dead-17>
Thu Oct 28 2004 11:17:46,0,m..,-rwxrwxrwx,0,0,16,<sanswork-_ap.gif-dead-16>
```

Now, mactime displays the timeline of files from a given input file. Ils and fls files may be used to run mactime against. This has given us a timeline of file activity within the image file pertaining to deleted files. Next we run fls to gather all of the filename information:

```
[root@LinuxForensics images]# fls -f fat -r sanswork
r/r 3: her.doc
r/r 4: hey.doc
r/r * 7: WinPcap_3_1_beta_3.exe (_INPCA~1.EXE)
r/r * 10: WinPcap_3_1_beta_3.exe (_INPCA~1.EXE)
r/r * 12: WinDump.exe (_INDUMP.EXE)
r/r * 14: WinDump.exe (_INDUMP.EXE)
r/r * 15: _apture
r/r * 16: _ap.gif
r/r * 17: _ap.gif
r/r 18: coffee.doc
```

fls allows us to view files within an image. This program works at the file name layer by listing allocated and deleted files. The file name layer of a file system is used primarily to assign human readable names to a file. It is far more efficient and easier to recall a directory and filename rather than it's physical location on the media, i.e. the inode, block, ect. This will also display recently deleted files as well. Well, we can see from the output of a simple fls command that there are at least seven deleted files (as indicated by the "*" in the filename field). The files **WinPcap_3_1_beta_3.exe**, and **WinDump.exe** are of interest here. These are windows programs used for packet capturing, but we will get into that later. Now, this image is getting interesting.

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)

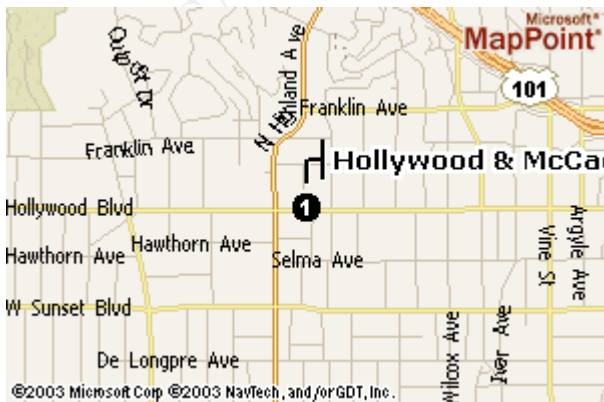
```
-----
511-550 (40) -> EOF
551-590 (40) -> EOF
591-630 (40) -> EOF
```

Now that we can see that there are deleted file, we will now move on and utilize Autopsy to attempt a recovery of these files. Autopsy is the graphical front end to the sleuth kit. It is a very intuitive application that makes using the sleuth kit tools much more efficient. Within Autopsy, I have selected the deleted files view, and clicked on the sectors to recover the files. The files that were recovered are:

_apture
_ap.gif
Windump.exe

Upon recovering, I inferred the names of the recovered files and named these files capture, map.gif, and windump.exe respectively. Upon scanning the files with Antivirus, no viruses were detected.

The map.gif file is a graphical file, which was indicated by a file command, and which most likely contains a map. Now, we will open this file using a graphics viewer and see what it contains.



We see that the file contains a picture of a map. This location must have been important for the suspect to know. We will move on and put everything together after all evidence is collected.

Knowing that windump.exe is a windows port of the Unix program tcpdump, I can infer that the file labeled "capture" is a network capture file. However, any file can be named anything you want it to be, so, I will execute the file command on the file labeled windump.exe, winpcap_3_1_beta4.exe and capture.

file WinDump.exe

WinDump.exe: MS-DOS executable (EXE), OS/2 or MS Windows

file WinPcap_3_1_beta4.exe

WinPcap_3_1_beta4.exe: MS-DOS executable (EXE), OS/2 or MS Windows

file capture

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

The file command has shown us that this file is indeed a packet capture file, based on the tcpdump binary. We also see that windump.exe and winpcap_3_1_beta4.exe are MS-DOS executables; these programs may be executed in a DOS or Windows environment. We can infer from this information that the suspect has performed actions on a Windows machine, or a machine which supports MS-DOS executable binaries. Now we will want to view the tcpdump capture file, to see exactly what network traffic the suspect was interested in. I have loaded that capture file into ethereal, a graphical network protocol analyzer, to view the packets which have been captured. This will break down any protocols, such as TCP, UDP, ect and display the information about the packets. The capture file seems to be focused on traffic generated from an internal IP address of 192.168.2.104 (private, non-routable IP) and seems to be directed at the destination IP 64.4.34.250. The 64.4.34.250 appears to be the IP for MS Hotmail, a free, web based mail store. We want to view the registration information on this address, so we will look this information up using samspade.org. This will provide the registration information pertaining to this IP. Here is the registration information:

Server Used: [whois.arin.net]

64.4.34.250 = [www.bay12.hotmail.com]

OrgName: MS Hotmail

OrgID: MSHOTM

Address: One Microsoft Way

City: Redmond

StateProv: WA

PostalCode: 98052
Country: US
NetRange: [64.4.0.0](#) - [64.4.63.255](#)
CIDR: 64.4.0.0/18
NetName: HOTMAIL
NetHandle: [NET-64-4-0-0-1](#)
Parent: NET-64-0-0-0-0
NetType: Direct Assignment
NameServer: [NS1.HOTMAIL.COM](#)
NameServer: [NS3.HOTMAIL.COM](#)
NameServer: [NS2.HOTMAIL.COM](#)
NameServer: [NS4.HOTMAIL.COM](#)
Comment:
RegDate: 1999-11-24
Updated: 2003-06-27
TechHandle: [MSFTP-ARIN](#)
TechName: MSFT-POC
TechPhone: 1-425-882-8080
TechEmail: iprrms@microsoft.com

OrgAbuseHandle: [ABUSE231-ARIN](#)
OrgAbuseName: Abuse
OrgAbusePhone: 1-425-882-8080
OrgAbuseEmail: abuse@microsoft.com

OrgTechHandle: [MSFTP-ARIN](#)
OrgTechName: MSFT-POC
OrgTechPhone: 1-425-882-8080
OrgTechEmail: iprrms@microsoft.com

So, Mr. Lawrence seems to be monitoring network traffic, specifically someone's email. It appears that he has captured the entire conversation, as we can see by the initial three way hand shake, the SYN, SYN-ACK, and ACK. I have looked over the entire capture file, and here is what I have found of interest. It looks like Ms. Conlay has emailed a friend, a Sam Guarillo, and is arranging a coffee meeting at the corner of Hollywood and McCadden St. Remember our mappoint map.gif file from above? This is the location that was targeted in that file. This appears to be how Mr. Lawrence knew where Ms. Conlay was going to be. I have extracted the plain text of this packet to make it easier to read:

From: flowergirl96@hotmail.com (Leila)
To: SamGuarillo@hotmail.com
subject= coffee

Sure coffee sounds great. Lets meet at the coffee shop on the corner Hollywood and McCadden. A nice out of the way spot. See you at 7pm - Leila

curmbox=F000000001&HrsTest=&_HMaction=Send&FinalDest=&subaction=&plaintext=&login=flowergirl96&msg=&start=&len=&attfile=&attlistfile=&eurl=&type=&src=&ref=&ru=&msgdhrid=b16479b18beec291196189c78555223c_1098692452&RTEbgcolor=&encodedto=SamGuarillo@hotmail.com&encodedcc=&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-LeilaHTTP/1.1 100 Continue

So, it looks like Mr. Lawrence copied the winpcap, and windump programs to a jump drive, and installed it on Ms. Conlays computer. He then executed the packet capture utility windump, and captured Ms. Conlays network traffic, displaying her private email. Since Mr. Lawrence was able to perform such a task, he knew where Ms. Conlay was going, when she would be there, and with whom she was meeting.

IMAGE DETAILS

This image contained three Microsoft word files, as well as seven deleted files. Here, again, is the listing of those files provided by the fls command.

r/r 3: her.doc

r/r 4: hey.doc

r/r * 7: WinPcap_3_1_beta_3.exe (_INPCA~1.EXE)

r/r * 10: WinPcap_3_1_beta_3.exe (_INPCA~1.EXE)

r/r * 12: WinDump.exe (_INDUMP.EXE)

r/r * 14: WinDump.exe (_INDUMP.EXE)

r/r * 15: _apture

r/r * 16: _ap.gif

r/r * 17: _ap.gif

r/r 18: coffee.doc

Most of the true names of the files are listed in the output above. The programs Mr. Lawrence has utilized to perform his crimes are the MS-DOS executable files:

WinPcap_3_1_beta_3.exe

WinDump.exe

The files that were saved are to the disk are:

Capture

Map.gif

Coffee.doc

Hey.doc

Her.doc

MACTIMES

The mactime utility will give us a timeline of the files, when they were created, when they were modified, and when they were last accessed.

```
[root@LinuxForensics images]# mactime -b sanswork.ils -h -d
```

The Sleuth Kit mactime Timeline
Input Source: sanswork.ils

Date,Size,Type,Mode,UID,GID,Meta,File Name

```
Wed Oct 27 2004 00:00:00,0,.a.,-rwxrwxrwx,0,0,12,<sanswork-_INDUMP.EXE-dead-12>
Wed Oct 27 2004 00:00:00,0,.a.,-rwxrwxrwx,0,0,7,<sanswork-_INPCA~1.EXE-dead-7>
Wed Oct 27 2004 16:23:50,485810,m.,-rwxrwxrwx,0,0,10,<sanswork-_INPCA~1.EXE-dead-10>
Wed Oct 27 2004 16:23:54,485810,..c,-rwxrwxrwx,0,0,10,<sanswork-_INPCA~1.EXE-dead-10>
Wed Oct 27 2004 16:23:54,0,..c,-rwxrwxrwx,0,0,7,<sanswork-_INPCA~1.EXE-dead-7>
Wed Oct 27 2004 16:23:56,0,m.,-rwxrwxrwx,0,0,7,<sanswork-_INPCA~1.EXE-dead-7>
Wed Oct 27 2004 16:24:02,450560,m.,-rwxrwxrwx,0,0,14,<sanswork-_INDUMP.EXE-dead-14>
Wed Oct 27 2004 16:24:04,450560,..c,-rwxrwxrwx,0,0,14,<sanswork-_INDUMP.EXE-dead-14>
Wed Oct 27 2004 16:24:04,0,..c,-rwxrwxrwx,0,0,12,<sanswork-_INDUMP.EXE-dead-12>
Wed Oct 27 2004 16:24:06,0,m.,-rwxrwxrwx,0,0,12,<sanswork-_INDUMP.EXE-dead-12>
Thu Oct 28 2004 00:00:00,53056,.a.,-rwxrwxrwx,0,0,15,<sanswork-_apture-dead-15>
Thu Oct 28 2004 00:00:00,8814,.a.,-rwxrwxrwx,0,0,17,<sanswork-_ap.gif-dead-17>
Thu Oct 28 2004 00:00:00,485810,.a.,-rwxrwxrwx,0,0,10,<sanswork-_INPCA~1.EXE-dead-10>
Thu Oct 28 2004 00:00:00,450560,.a.,-rwxrwxrwx,0,0,14,<sanswork-_INDUMP.EXE-dead-14>
Thu Oct 28 2004 00:00:00,0,.a.,-rwxrwxrwx,0,0,16,<sanswork-_ap.gif-dead-16>
Thu Oct 28 2004 11:08:24,53056,..c,-rwxrwxrwx,0,0,15,<sanswork-_apture-dead-15>
Thu Oct 28 2004 11:11:00,53056,m.,-rwxrwxrwx,0,0,15,<sanswork-_apture-dead-15>
Thu Oct 28 2004 11:17:44,0,..c,-rwxrwxrwx,0,0,16,<sanswork-_ap.gif-dead-16>
Thu Oct 28 2004 11:17:44,8814,..c,-rwxrwxrwx,0,0,17,<sanswork-_ap.gif-dead-17>
Thu Oct 28 2004 11:17:46,8814,m.,-rwxrwxrwx,0,0,17,<sanswork-_ap.gif-dead-17>
Thu Oct 28 2004 11:17:46,0,m.,-rwxrwxrwx,0,0,16,<sanswork-_ap.gif-dead-16>

Mon Oct 25 2004 00:00:00      19968 .a. -/-rwxrwxrwx 0          0          3
D:\her.doc
Mon Oct 25 2004 08:32:06      19968 ..c -/-rwxrwxrwx 0          0          3
D:\her.doc
Mon Oct 25 2004 08:32:08      19968 m.. -/-rwxrwxrwx 0          0          3
D:\her.doc
Tue Oct 26 2004 00:00:00      19968 .a. -/-rwxrwxrwx 0          0          4
D:\hey.doc
Tue Oct 26 2004 08:48:06      19968 ..c -/-rwxrwxrwx 0          0          4
D:\hey.doc
Tue Oct 26 2004 08:48:10      19968 m.. -/-rwxrwxrwx 0          0          4
D:\hey.doc
Wed Oct 27 2004 00:00:00      450560 .a. -/-rwxrwxrwx 0          0          12
D:\WinDump.exe (_INDUMP.EXE) (deleted)
                                0 .a. -rwxrwxrwx 0          0          12
<sanswork-_INDUMP.EXE-dead-12>
                                485810 .a. -/-rwxrwxrwx 0          0          7
D:\WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
                                0 .a. -rwxrwxrwx 0          0          7
<sanswork-_INPCA~1.EXE-dead-7>
Wed Oct 27 2004 16:23:50      485810 m.. -rwxrwxrwx 0          0          10
<sanswork-_INPCA~1.EXE-dead-10>
```



```

485810 m.. -/-rwxrwxrwx 0 0 10
D:\WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
Wed Oct 27 2004 16:23:54 485810 ..c -rwxrwxrwx 0 0 10
<sanswork-_INPCA~1.EXE-dead-10>
0 ..c -rwxrwxrwx 0 0 7
<sanswork-_INPCA~1.EXE-dead-7>
485810 ..c -/-rwxrwxrwx 0 0 7
D:\WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
485810 ..c -/-rwxrwxrwx 0 0 10
D:\WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
Wed Oct 27 2004 16:23:56 485810 m.. -/-rwxrwxrwx 0 0 7
D:\WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
0 m.. -rwxrwxrwx 0 0 7
<sanswork-_INPCA~1.EXE-dead-7>
Wed Oct 27 2004 16:24:02 450560 m.. -rwxrwxrwx 0 0 14
<sanswork-_INDUMP.EXE-dead-14>
450560 m.. -/-rwxrwxrwx 0 0 14
D:\WinDump.exe (_INDUMP.EXE) (deleted)
Wed Oct 27 2004 16:24:04 0 ..c -rwxrwxrwx 0 0 12
<sanswork-_INDUMP.EXE-dead-12>
450560 ..c -/-rwxrwxrwx 0 0 14
D:\WinDump.exe (_INDUMP.EXE) (deleted)
450560 ..c -rwxrwxrwx 0 0 14
<sanswork-_INDUMP.EXE-dead-14>
450560 ..c -/-rwxrwxrwx 0 0 12
D:\WinDump.exe (_INDUMP.EXE) (deleted)
Wed Oct 27 2004 16:24:06 450560 m.. -/-rwxrwxrwx 0 0 12
D:\WinDump.exe (_INDUMP.EXE) (deleted)
0 m.. -rwxrwxrwx 0 0 12
<sanswork-_INDUMP.EXE-dead-12>
Thu Oct 28 2004 00:00:00 450560 .a. -rwxrwxrwx 0 0 14
<sanswork-_INDUMP.EXE-dead-14>
8814 .a. -/-rwxrwxrwx 0 0 17
D:\_ap.gif (deleted)
0 .a. -rwxrwxrwx 0 0 16
<sanswork-_ap.gif-dead-16>
485810 .a. -/-rwxrwxrwx 0 0 10
D:\WinPcap_3_1_beta_3.exe (_INPCA~1.EXE) (deleted)
53056 .a. -rwxrwxrwx 0 0 15
<sanswork-_apture-dead-15>
8814 .a. -rwxrwxrwx 0 0 17
<sanswork-_ap.gif-dead-17>
485810 .a. -rwxrwxrwx 0 0 10
<sanswork-_INPCA~1.EXE-dead-10>
450560 .a. -/-rwxrwxrwx 0 0 14
D:\WinDump.exe (_INDUMP.EXE) (deleted)
53056 .a. -/-rwxrwxrwx 0 0 15
D:\_apture (deleted)
8814 .a. -/-rwxrwxrwx 0 0 16
D:\_ap.gif (deleted)
19968 .a. -/-rwxrwxrwx 0 0 18
D:\coffee.doc
Thu Oct 28 2004 11:08:24 53056 ..c -rwxrwxrwx 0 0 15
<sanswork-_apture-dead-15>
53056 ..c -/-rwxrwxrwx 0 0 15
D:\_apture (deleted)

```

```

Thu Oct 28 2004 11:11:00    53056 m.. -/-rwxrwxrwx 0      0      15
D:\/_apture (deleted)
                               53056 m.. -rwxrwxrwx 0      0      15
<sanswork-_apture-dead-15>
Thu Oct 28 2004 11:17:44      0 ..c -rwxrwxrwx 0      0      16
<sanswork-_ap.gif-dead-16>
                               8814 ..c -/-rwxrwxrwx 0      0      17
D:\/_ap.gif (deleted)
                               8814 ..c -/-rwxrwxrwx 0      0      16
D:\/_ap.gif (deleted)
                               8814 ..c -rwxrwxrwx 0      0      17
<sanswork-_ap.gif-dead-17>
Thu Oct 28 2004 11:17:46    8814 m.. -/-rwxrwxrwx 0      0      17
D:\/_ap.gif (deleted)
                               8814 m.. -/-rwxrwxrwx 0      0      16
D:\/_ap.gif (deleted)
                               8814 m.. -rwxrwxrwx 0      0      17
<sanswork-_ap.gif-dead-17>
                               0 m.. -rwxrwxrwx 0      0      16
<sanswork-_ap.gif-dead-16>
Thu Oct 28 2004 19:24:46   19968 ..c -/-rwxrwxrwx 0      0      18
D:\coffee.doc
Thu Oct 28 2004 19:24:48   19968 m.. -/-rwxrwxrwx 0      0      18
D:\coffee.doc

```

The mactime utility provides a visual representation of the attributes associated with the timeline of a file. These attributes include the Modify time, Access time, and Creation Time of a file. This information is extremely valuable in an computer investigation. This is how we gather timeline information, which will allow us to pinpoint dates and times that a computer crime was committed. This information is very important to the prosecution of a computer crime case, as they can prove when the activities have taken place.

As we can see from the mactimes, these files have been modified and accessed several times. The first document, her.doc, appears to have been created at 0832 on Mon. Oct. 25th 2004. The next document, hey.doc, was created almost exactly 24 hours later, at 0848 on Tue. Oct. 26th 2004. The file to next be utilized is the winpcap program. This is required by the windump program. Although this file has been deleted, it appears that from the timeline that the program was last utilized at 1623 on Oct. 27th, this is most likely when the file was deleted from the jumpdrive. FAT file systems do not set the Deleted time attribute as UNIX systems do. The attribute will be displayed as the C time. Next, windump.exe has a timestamp of 1624 on Oct. 27th, which, again, appears to be the deletion time of the file from the jumpdrive. Now, we know from our analysis up to this point that the network traffic was captured to the file named capture. This file has a timestamp of 1108 on Oct. 28th. So, it appears that windump.exe was deleted on the 27th, the suspect then performed analysis on the capture file, and then deleted the capture file on the 28th. To get the true mactimes as to when the programs were executed on the victim machine, an in depth forensic analysis on Ms. Conolays machine would be in order. Now, at 1117 on Oct. 28th, the map.gif file was deleted. This makes sense to us, since the suspect now had the

network traffic in hand, and could map out the location the victim was going to be from reading her email. On Oct. 28th at 1924, the coffee.doc file was created, and modified a few seconds later (file being saved). This is the document that appears to be threatening to Ms. Conolay, and follows in suit with the timeline of events.

We can see from the mactime output that the suspect had deleted evidence from the jumpdrive. This evidence has been recovered, and analyzed in other sections of this practical. The timeline of events begin on Oct. 25th, 2004, and end on Oct. 28th, 2004.

Forensic Details

Mr. Lawrence appears to have utilized the program windump.exe, which is a Windows port of popular UNIX TCPDUMP program. Windump is a packet capturing utility to be used on a Microsoft Windows platform machine to capture network traffic. Additional information regarding windump can be found at:

From <http://windump.polito.it/default.htm>

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

In addition to the windump program, there was also evidence of the WinPcap library, a program which is needed by windump.exe. Without this software installed, you will not be able to capture packets using windump.

<http://winpcap.polito.it/>

“WinPcap is an open source library for packet capture and network analysis for the Win32 platforms. It includes a kernel-level packet filter, a low-level dynamic link library (packet.dll), and a high-level and system-independent library (wpcap.dll, based on libpcap version 0.6.2).

The packet filter is a device driver that adds to Windows 95, 98, ME, NT, 2000, XP and 2003 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 capture primitives that are compatible with libpcap, the well known Unix capture library. These functions allow to

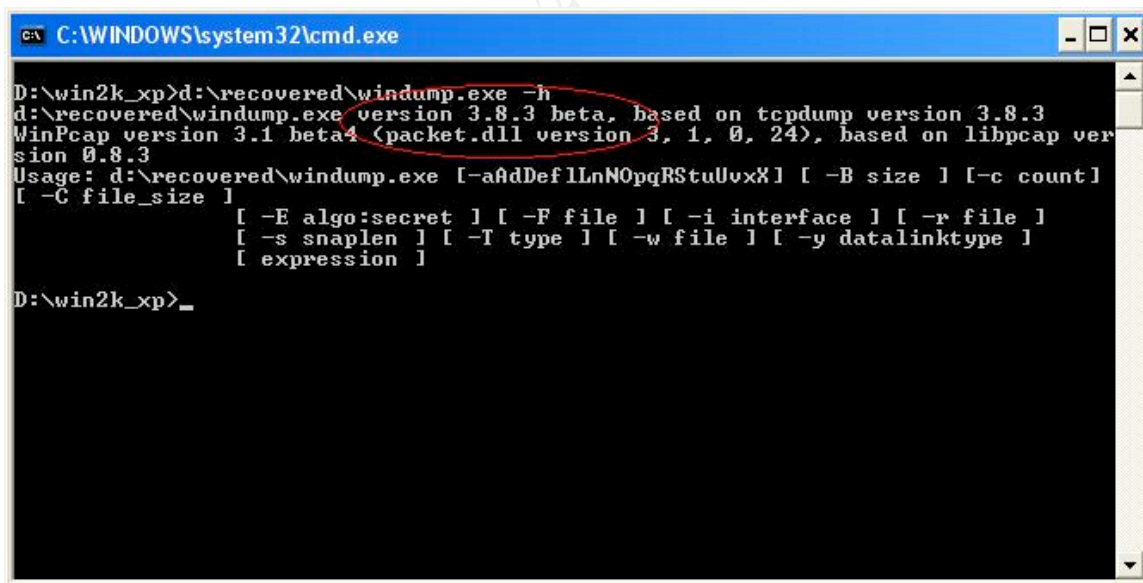
capture packets in a way independent from the underlying network hardware and operating system.”

The windump program puts the computers network card into promiscuous mode, which enables the program to capture network traffic traveling over the wire. Windump has many options that may be passed to it. One of the switches that appear to have been used in this case is the `-w` switch. This option instructs windump to write the output to a file, in this case, the file was aptly named capture. The file that is created with the `-w` switch can be loaded by windump with the `-r` switch, or using other network sniffing programs, such as ethereal, for analysis.

The program windump seems to have been executed on Wed Oct 27 2004 @16:24:02. Just one min. before that, evidence from the mactime output indicates that winpcap was executed, which makes sense, since winpcap needs to be installed before running windump.

Program Identification

The programs that have been identified are windump and winpcap. These programs are widely available and used throughout the IT community. Since these are well known programs, we can easily find and download the programs from the internet. There is no compiling or re-compiling necessary.



```
C:\WINDOWS\system32\cmd.exe
D:\win2k_xp>d:\recovered\windump.exe -h
d:\recovered\windump.exe version 3.8.3 beta, based on tcpdump version 3.8.3
WinPcap version 3.1 beta4 (packet.dll version 3, 1, 0, 24), based on libpcap ver
sion 0.8.3
Usage: d:\recovered\windump.exe [-aAdDeflLnNOpqRStuUvwxX] [-B size] [-c count]
[-C file_size]
      [-E algo:secret] [-F file] [-i interface] [-r file]
      [-s snaplen] [-T type] [-w file] [-y datalinktype]
      [expression]
```

To verify the program, I have downloaded windump.exe from <http://windump.polito.it/install/default.htm> to the root of c:\ on my Windows XP machine. From there, I executed the windump.exe command with the `-h` switch so that we can view the version of windump.

```
C:\WINDOWS\system32\cmd.exe

C:\>WinDump.exe -h
WinDump.exe version 3.8.3 beta, based on tcpdump version 3.8.3
WinPcap version 3.1 beta4 (packet.dll version 3, 1, 0, 24), based on libpcap version 0.8.3
Usage: WinDump.exe [-aAdDeflLnNOpqRStuUvxxX] [-B size] [-c count] [-C file_size]
                  [-E algo:secret] [-F file] [-i interface] [-r file]
                  [-s snaplen] [-T type] [-w file] [-y datalinktype]
                  [expression]

C:\>_
```

As you can see, the file versions are the same, version 3.8.3 beta of WinDump.exe. We will now move on and check the integrity of the file, let's compare the MD5 hash:

```
C:\WINDOWS\system32\cmd.exe

D:\win2k_xp>md5sum d:\recovered\windump.exe
79375b77975aa53a1b0507496107bff7 *d:\recovered\windump.exe

D:\win2k_xp>md5sum c:\windump.exe
79375b77975aa53a1b0507496107bff7 *c:\windump.exe

D:\win2k_xp>_
```

Ok, we can see that the original file that was recovered and copied to d:\recovered and the freshly downloaded file located on the root of c: both have the same MD5 value. It is safe to say that this is the exact program that was used to perform the actions in question, since the output of md5sum shows us the same hash value.

Legal Implications

Clearly, Mr. Lawrence has illegally monitored the electronic transmissions (network traffic in this case) of Ms. Leila Conlay. In doing so, he has violated the Federal Wiretap act; the law put forth by the government making it illegal to monitor the electronic transmissions of individuals and corporations. The act is discussed in more detail at <http://www.cybercrime.gov/usc2511.htm>

The wiretap act prohibits the interception, use, or disclosure of wire and electronic communications unless a statutory exception applies. This is fundamentally an exception to the rule or law. The federal wiretap act includes many exceptions; however none seem to apply to this situation as I see it. I do not believe Ms. Conolay consented to allowing her network traffic to be monitored by this individual, or wire “sniffing”, nor do I believe that Mr. Lawrence is a federal official who has been granted the right to sniff the traffic, nor does it seem that he “owns” the network for which Ms. Conolay is a user of.

As well as violating the federal wiretap act, it is my contention that Mr. Lawrence has also violated the Electronic Communications Privacy Act (ECPA) as well. With the installation of a packet sniffer, Mr. Lawrence had captured Ms. Conolay’s network traffic, and in doing so attained the ability to read Ms. Conolay’s private email, which is stored at hotmail.com. This act has clearly violated the individual’s privacy, allowing the suspect to be privy to personal information which Ms. Conolay would normally not divulge to Mr. Lawrence.

These laws that are protected by the Govt. have penalties associated with them. In violating these laws, these acts have a penalty of a fine and/or a maximum of five years imprisonment.

Recommendations

It would be my recommendation to the security administrator to ensure that the machines on the network adhere to stringent security policies. Users should not be able to install software, especially software that manipulates the TCP configuration of a machine. Users should not be able place a network card into promiscuous mode for any reason.

For further investigation on this matter, I would have the security administrator review all logs on Ms. Conolay’s computer. Firewall and IDS logs could also play a helpful role in this investigation. We would want to see if the suspect was opening any connections to Ms. Conolay’s email after the fact, as well as other resources she may possess, i.e. bank information, credit card numbers.

I believe that Ms. Conolay’s computer should be imaged, and thoroughly investigated. We would want to look at MAC times, new files, deleted files, as well as programs that do not belong. We would want to look for any configuration

changes, as well as backdoors that may have been installed. Also, we would be looking for evidence of keystroke loggers that may have been installed, both software and hardware, the tiny device that plugs into the back of the computer. A thorough investigation of the machine in question could prove if the programs Winpcap and windump were ever placed on Ms. Conolay's computer.

It is clear from the investigation that the suspect in question has violated the law pertaining to computer crimes, as well as violating another individual's privacy.

Additional Information

<http://www.surasoft.com/articles/packetsniffing.php>

Excellent write up on packet sniffing, as well as ways to protect yourself.

<http://www.computerforensics.net/forensics.htm>

This is a useful article on computer forensics in general. It includes information on business processes pertaining to computer forensics.

<http://www.cftt.nist.gov/>

Interesting site that provides information regarding tools that pertain to the vast field of computer forensics.

References

Gerald Combs, original author <http://www.ethereal.com/>

Sleuth Kit Forensic toolset homepage <http://www.sleuthkit.org/sleuthkit/index.php>

Hudak, Tyler "Analysis of a Linux honeypot"

http://www.giac.org/practical/GCFA/Tyler_Hudak_GCFA.pdf

Degioanni, Loris Main Programmer for windump. Software developed by the Lawrence Berkley Laboratory Network Research Group. <http://www-nrg.ee.lbl.gov/>, <http://windump.polito.it/>

Degioanni, Loris Main Programmer for winpcap. Software developed by the Lawrence Berkley Laboratory Network Research Group. <http://winpcap.polito.it/>

Computer Forensics Tool Testing (CFTT) Project Homepage.

<http://www.cftt.nist.gov/>

Carrier, Brian "File system analysis techniques"

http://www.sleuthkit.org/sleuthkit/docs/ref_fs.html,

Carrier, Brian "File activity timelines"

http://www.sleuthkit.org/sleuthkit/docs/ref_timeline.html

Lucas, Charles "Running Sleuthkit and Autopsy under Windows"

http://www.memophase.net/Running_Sleuthkit_and_Autopsy_Under_Windows.pdf

18 U.S.C. 2511. Interception and Disclosure of Wire, Oral, or Electronic Communications Prohibited <http://www.cybercrime.gov/usc2511.htm>

Ryder, Karen "Computer Forensics – We've had an incident, who do we get to investigate?" GSEC certification assignment

<http://www.sans.org/rr/whitepapers/incident/652.php>

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