GIAC Certified Forensic Analyst (GCFA) Practical Assignment Version 1.4 (July 21, 2003)

By Steven Hickey

Submitted on December 16, 2003

Summary

This paper contains the practical assignment of the GIAC Certified Forensics Analyst program. This practical exercise is designed to demonstrate my understanding of the course material prior to testing.

The assignment has 3 parts:

Analyze an Unknown Binary – In this part, I downloaded a file containing a binary file that was part of an internal corporate investigation into employee wrongdoing. Since I was not involved in the data acquisition I must rely on information provided by the response team. It's my job to verify authenticity of my copy and perform analysis of the binary to determine it's function, how it relates to alleged activates and what if any impact its use may have on our corporate liability.

Perform Forensic Tool Validation – In legal proceedings, opposition counsel can question tools used during investigation. In this part of the assignment I test the operation of a handheld disk duplication device by comparing its operation to other industry-standard disk duplication methods. The testing and validation demonstrates an understanding of the processes involved and provides documentation support if I'm ever questioned about the tool.

Legal Issues of Incident Handling – This part includes research of some legal issues that a forensics investigator may encounter. While we are not attorneys, we should operate under the assumption that every case is going to court. Therefore data handling must preserve, not destroy evidence. A basic understanding of the laws is essential to our effectiveness as investigators.

Part 1 -- Analyze an Unknown Binary

Background – Employee John Price has been suspended as a result of a random IT audit of computer usage. Audit findings indicate that Mr. Price has used the organization's resources to illegally distribute copyrighted materials. Prior to the Incident Response Team's (IRT) arrival Mr. Price was able to wipe his hard drive, erasing any useful evidence. However, IRT found a single 3.5" floppy disk in the drive of Mr. Price's PC. The floppy was cataloged and taken into evidence even though Mr. Price denied that it belonged to him. The assigned task for this project is to analyze a binary file named "prog" located on the floppy disk, establish its purpose and how the suspect might have used it in his alleged illegal activities. Also, I will examine the diskette for other relevant evidence.

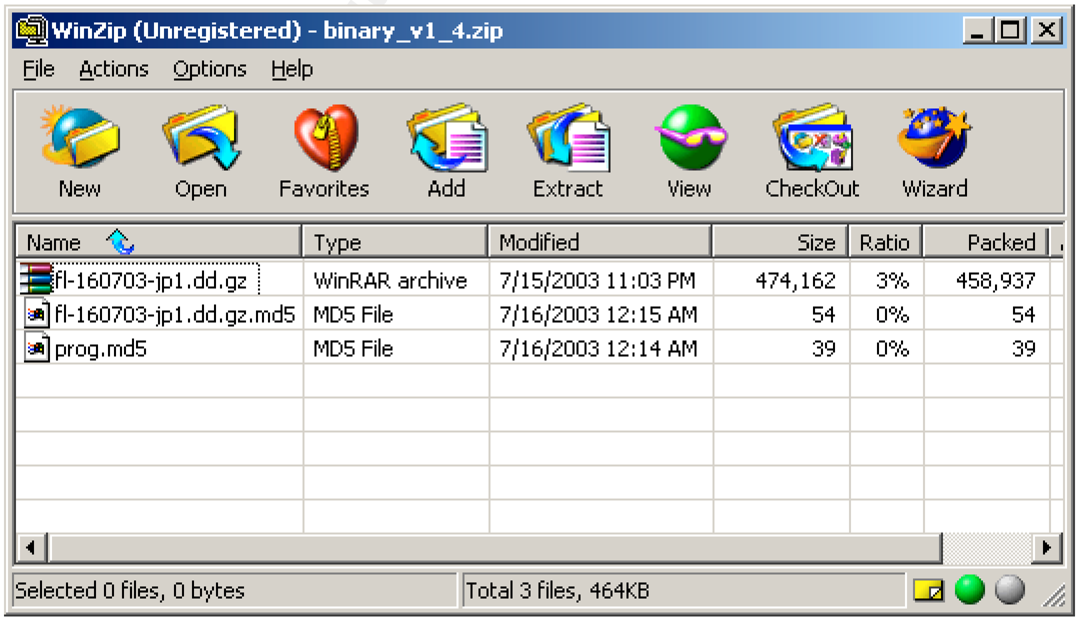
In this case, the Incident Response Team has assumed the responsibility for chain of custody. They have noted date and time, location, equipment make, model, and serial number along with names of the individual parties and detailed descriptions of all evidence involved. Also they have secured the evidence according to procedure.

I have received a zipped file from the IRT via email with the following description.

Tag# fl-160703-jp1
3.5 inch TDK floppy disk
MD5: 4b680767a2aed974cec5fbcbf84cc97a
fl-160703-jp1.dd.gz

The zipped file contains 3 files:

fl-160703-jp1.dd.gz
fl-160703-jp1.dd.gz.md5
prog.md5



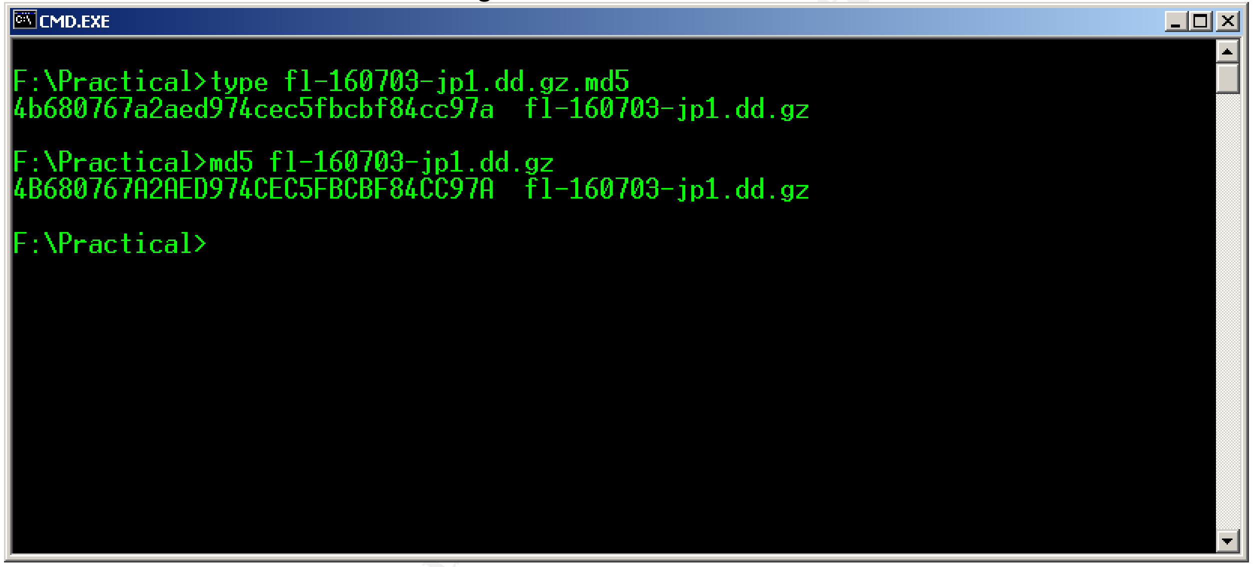
Name	Type	Modified	Size	Ratio	Packed
fl-160703-jp1.dd.gz	WinRAR archive	7/15/2003 11:03 PM	474,162	3%	458,937
fl-160703-jp1.dd.gz.md5	MD5 File	7/16/2003 12:15 AM	54	0%	54
prog.md5	MD5 File	7/16/2003 12:14 AM	39	0%	39

Selected 0 files, 0 bytes
Total 3 files, 464KB

Once extracted, I find that the 2 files with .MD5 extensions are text containing the MD5 hashes for the files, fl-160703-jp1.dd.gz and prog respectively.

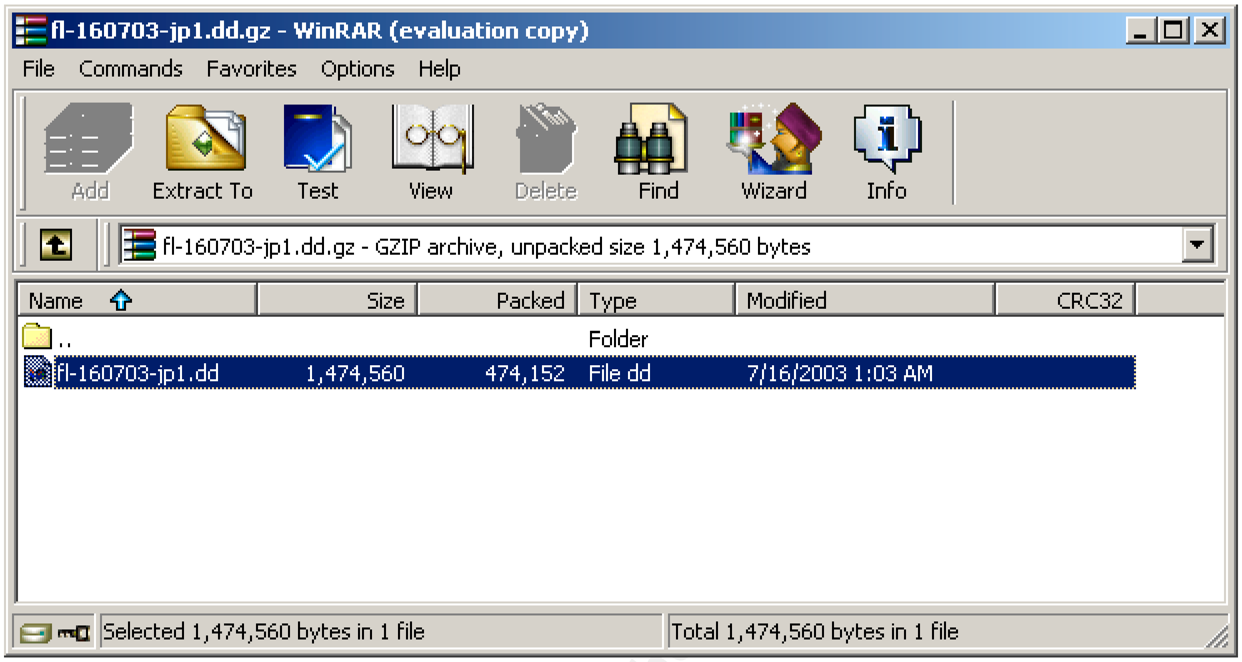
> MD5 or Message-Digest 5 is a mathematical algorithm that generates a non-reversible numeric representation of a message. In this case the message is the GZIPed, DD image file captured from the floppy diskette in question. The resulting 32-digit hexadecimal number called a hash is so unique, as to provide an accepted "fingerprint" of the processed message. It's widely accepted that it is "computationally infeasible" for 2 different messages to result in the same MD5 hash value.

Because the contents of the fl-160703-jp1.dd.gz.md5 text file and a live MD5 hash on the physical fl-160703-jp1.dd.gz file are identical, I can be assured that the file sent in the ZIP is identical to the file that generated the hash value.



```
F:\Practical>type fl-160703-jp1.dd.gz.md5
4b680767a2aed974cec5fbcbf84cc97a  fl-160703-jp1.dd.gz

F:\Practical>md5 fl-160703-jp1.dd.gz
4B680767A2AED974CEC5FBCBF84CC97A  fl-160703-jp1.dd.gz

F:\Practical>
```

The file fl-160703-jp1.dd.gz has arrived as a GNU Zip archive. WinRAR reveals the contents as one file named fl-160703-jp1.dd dated 7/16/2003 at 1:03 AM. The file extracts without error.


fl-160703-jp1.dd.gz - WinRAR (evaluation copy)
File Commands Favorites Options Help
Add Extract To Test View Delete Find Wizard Info

fl-160703-jp1.dd.gz - GZIP archive, unpacked size 1,474,560 bytes

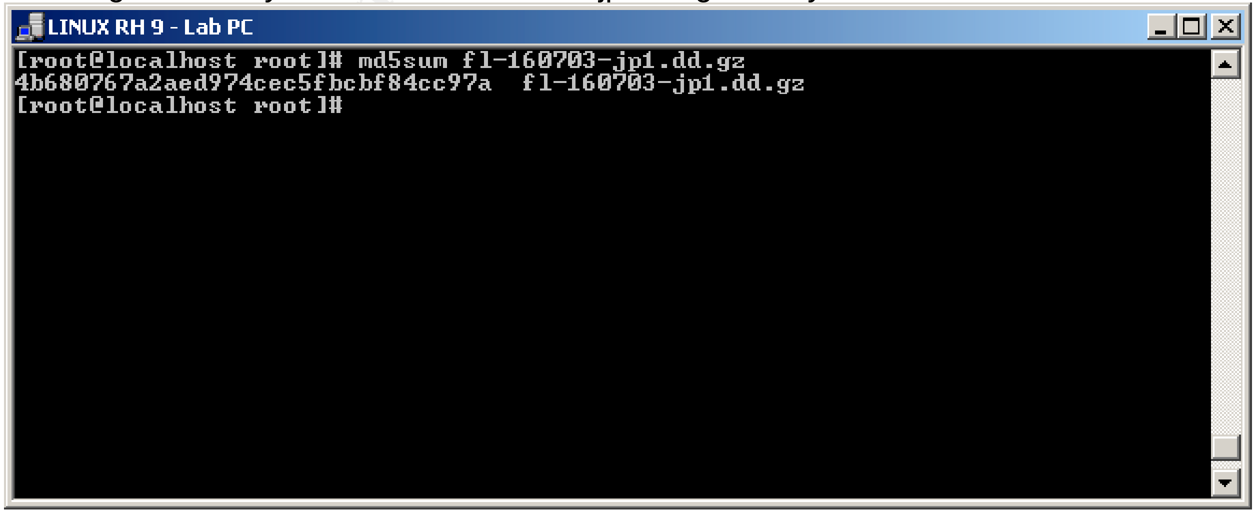
Name	Size	Packed	Type	Modified	CRC32
..			Folder		
fl-160703-jp1.dd	1,474,560	474,152	File dd	7/16/2003 1:03 AM	

Selected 1,474,560 bytes in 1 file

Total 1,474,560 bytes in 1 file

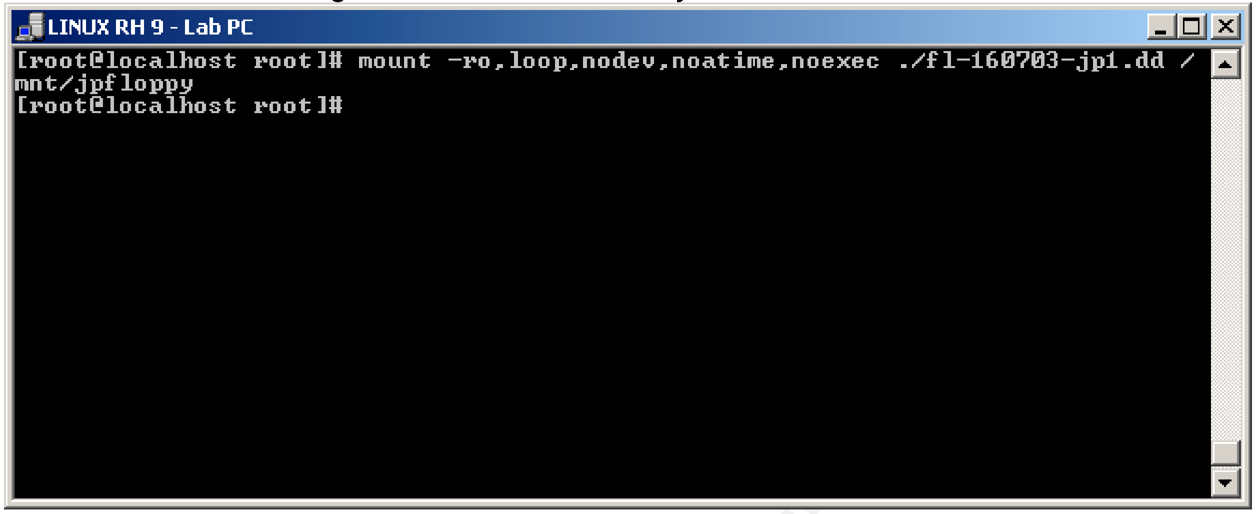
Now that I'm sure that the file has arrived intact, I move to my Linux forensics workstation. I'm running Linux RedHat v9. I use a clean installation of my forensics configuration on a sanitized hard drive. I have established procedures to sanitize the hard drive by performing a complete wiping of the drive then I restore an image of my forensics workstation configuration. This process prevents the possibility of cross-contamination between cases and begins all cases with the same set of tools addressing potential claims of favoritism from one case to another.

Once again, I verify the MD5 of fl-160703-jp1.dd.gz on my Linux workstation.



```
[root@localhost root]# md5sum fl-160703-jp1.dd.gz
4b680767a2aed974cec5fbcbf84cc97a  fl-160703-jp1.dd.gz
[root@localhost root]#
```

I extracted the DD image file from the fl-160703-jp1.dd.gz. Then, to safely work with the contents of the DD image, I mount it in read-only, non-execute mode as follows.

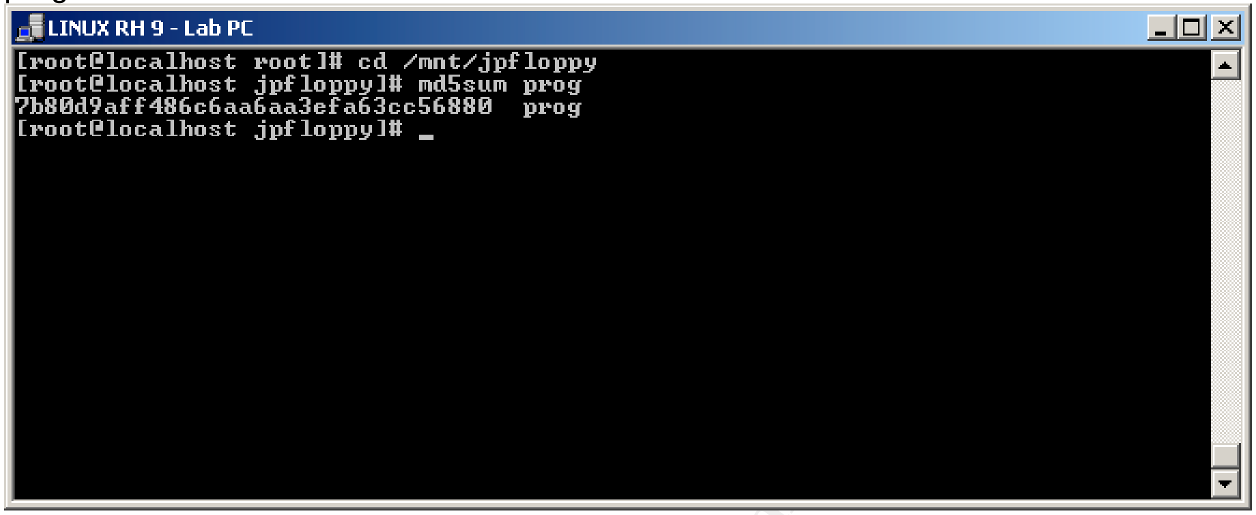


```
[root@localhost root]# mount -ro,loop,nodev,noatime,noexec ./fl-160703-jp1.dd /
mnt/jpfloppy
[root@localhost root]#
```

Where mount options are:

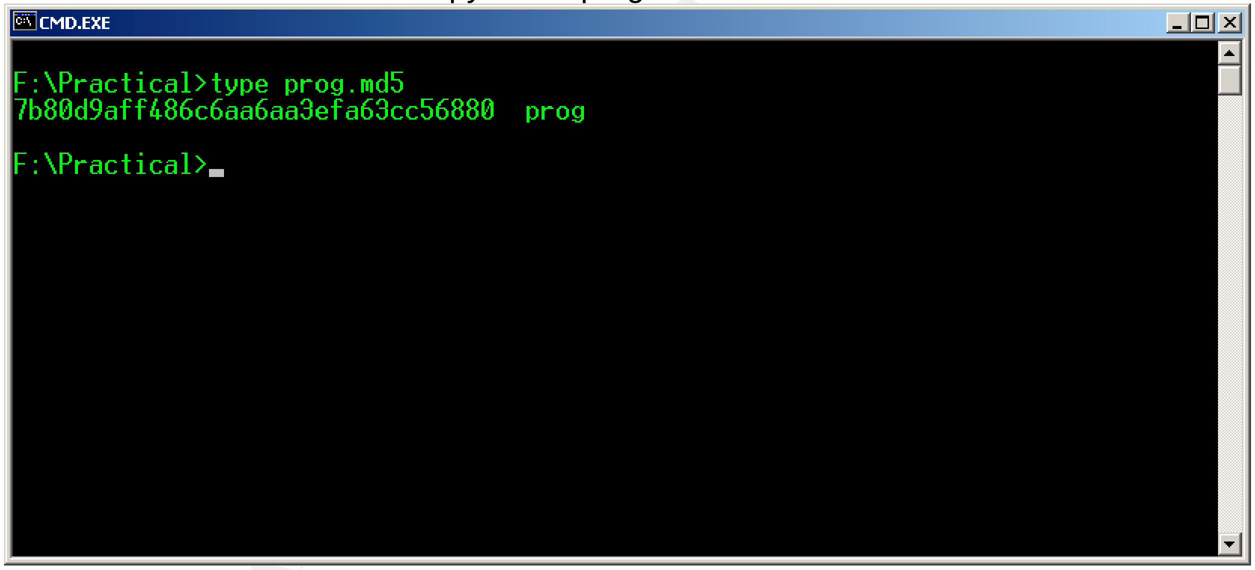
ro	Mount the file system read-only.
loop	Mount via the loop device (for image files)
nodev	Do not interpret character or block special devices on the file system.
noatime	Do not update inode access times on this file system (e.g, for faster access on the news spool to speed up news servers).
noexec	Do not allow execution of any binaries on the mounted file system. This option might be useful for a server that has file systems containing binaries for architectures other than its own.

I can now navigate to the mounted floppy drive image and verify the MD5 hash of the prog file.



```
[root@localhost root]# cd /mnt/jpfloppy
[root@localhost jpfloppy]# md5sum prog
7b80d9aff486c6aa6aa3efa63cc56880  prog
[root@localhost jpfloppy]# _
```

Since this MD5 matches the prog.md5 text file provided in the original zip file I am assured that I have an exact copy of the prog file.



```
F:\Practical>type prog.md5
7b80d9aff486c6aa6aa3efa63cc56880  prog

F:\Practical>_
```

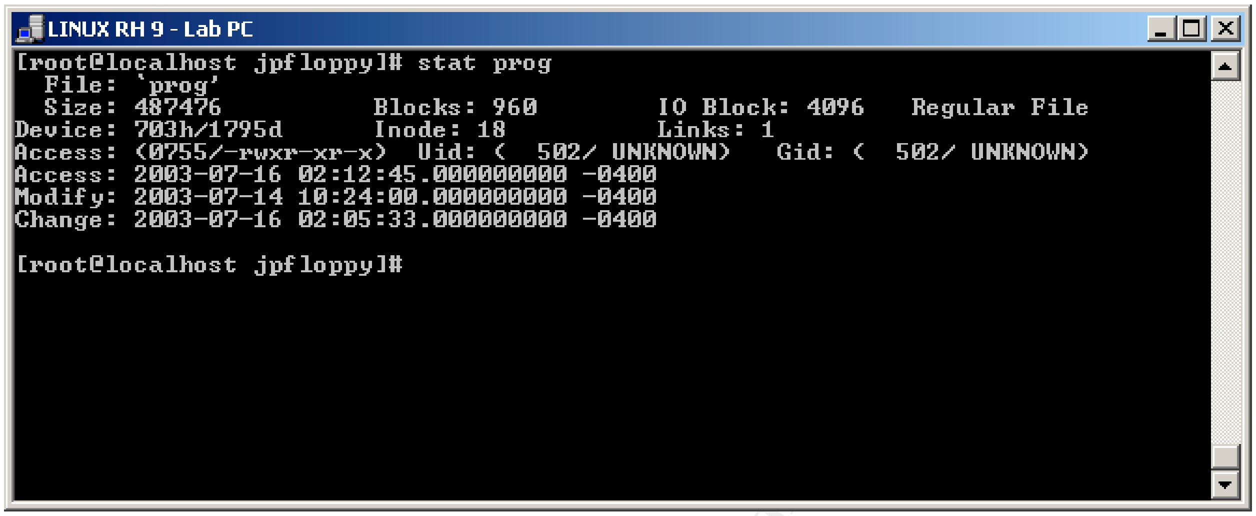
I have gone through the exercise of moving the evidence file from my Windows workstation to the Linux forensics workstation for 2 reasons.

1. In the scenario provided in the assignment I received the file via download or most likely email. Under ideal circumstances, forensics workstations are prohibited from being connected to the Internet or other network-connected devices. So I probably would have received the file on my network-attached workstation, a Windows PC.

2. The MD5 verification of files using both the Windows and Linux platforms satisfies a basic requirement of sound forensics analysis: tests and process must be both verifiable and repeatable.

Binary Details -- Next, I will perform some basic, static analysis of the actual file.

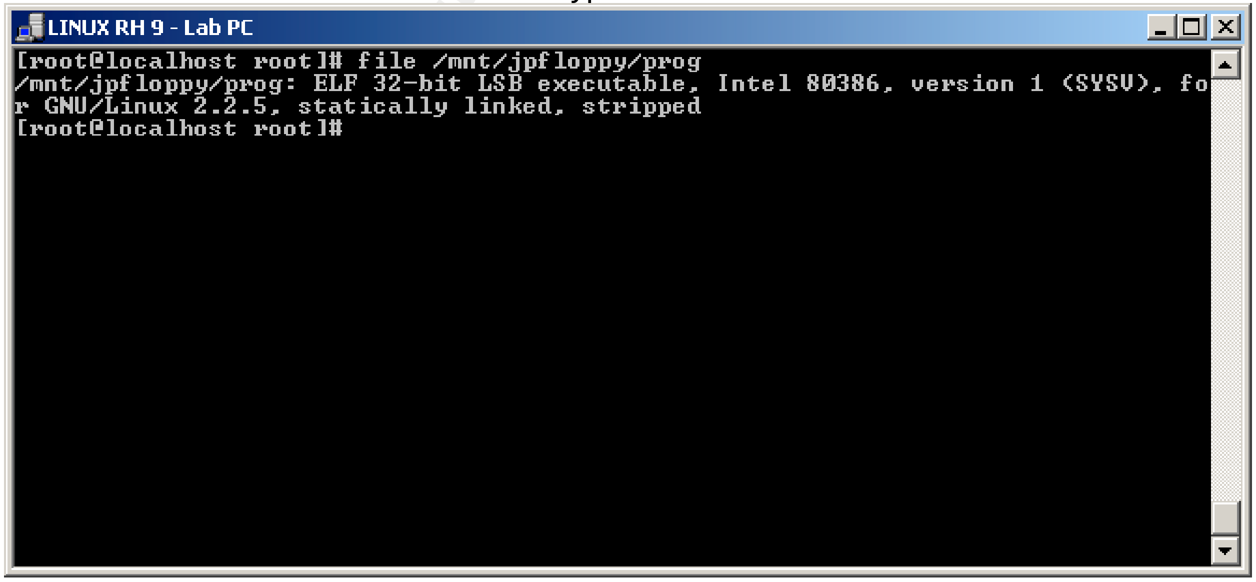
The STAT command reveals information about the file:



```
[root@localhost jpfloppy]# stat prog
  File: `prog'
  Size: 487476          Blocks: 960          IO Block: 4096   Regular File
Device: 703h/1795d      Inode: 18            Links: 1
Access: (0755/-rwxr-xr-x)  Uid: (  502/ UNKNOWN)    Gid: (  502/ UNKNOWN)
Access: 2003-07-16 02:12:45.000000000 -0400
Modify: 2003-07-14 10:24:00.000000000 -0400
Change: 2003-07-16 02:05:33.000000000 -0400

[root@localhost jpfloppy]#
```

File Size	487,476 bytes	
User ID	502	
Group ID	502	
Access Permissions	Owner = rwx (read, write and execute)	7
	Group = r-x (read and execute)	5
	Others = r-x (read and execute)	5
Last Accessed Time	02:12:45 on July 16, 2003	
Modified Time	10:24:00 on July 14, 2003	
Changed Time	02:05:33 on July 16, 2003	

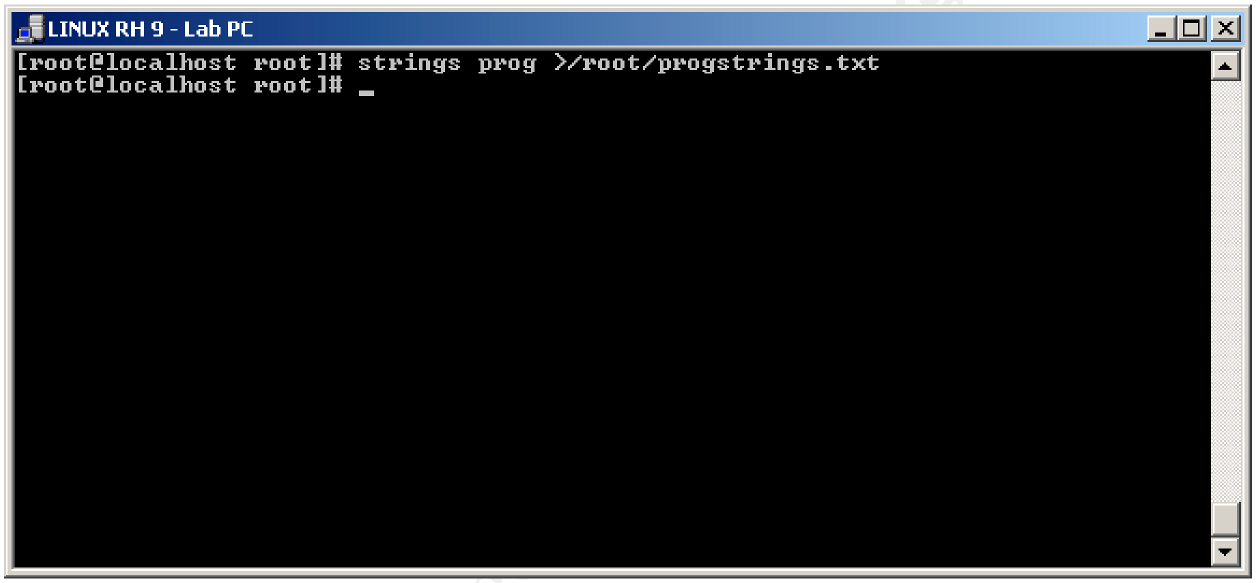
The FILE command determines a file's type.



```
[root@localhost root]# file /mnt/jpfloppy/prog
/mnt/jpfloppy/prog: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), fo
r GNU/Linux 2.2.5, statically linked, stripped
[root@localhost root]#
```

ELF stands for Executable and Linking Format, originally developed by Unix System Laboratories; it is an executable program file. We see from the FILE command output it's compiled to run on an Intel processor, it's statically linked and is stripped. Statically linked indicates that needed libraries are included inside the ELF file, this provides more portable code but generates a larger file. The stripped indication means the symbol table and debugging information is removed, this reduces the size of the code.

Program Description and Identification -- Now we know the file is a Linux executable program. Let's take a look "inside" the program. The main contents of executable files are program instruction codes, which are not readable. However, we can use the STRINGS command to output the text readable portions of the file. These text readable strings may give us some insight in to the purpose of the program.



```
[root@localhost root]# strings prog >/root/progstrings.txt
[root@localhost root]# _
```

Since the results of the STRINGS command yields almost 55,000 characters, I'll redirect the output to a file and open it up in a separate text file viewer.

We get our first hints at line number 713 with text references to "mft_getopt" then "mft_log_init", "MFT_LOG_THRESH", "mft_log_shutdown". Are these references to the Master File Table?

The next cluster of coherent text is "test for fragmentation (returns 0 if file is fragmented)", "checkfrag", "display fragmentation information for the file", and "frag". Do these refer to disk or file fragmentation?

Do the "wipe the file from the raw device" and "wipe" text mean something?

Here's something interesting: "1.0.20 (07/15/03)" is found at approximately line 848. The date is between our file modified and changed dates. Does "1.0.20" refer to a version number?

Oh, here's something "use block-list knowledge to perform special operations on files" There are several references to "slack". Does this program do something with file or disk slack space?

Searching for an @ character to see if there's an email address, yields the following section of text at approximately line number 4677

```
1997-12-20
+45 3325-6543
+45 3122-6543
keld@dkuug.dk
Keld Simonsen
ISO/IEC 14652 i18n FDCC-set
C/o Keld Simonsen, Skt. Jorgens Alle 8, DK-1615 Kobenhavn V
ISO/IEC JTC1/SC22/WG20 - internationalization
```

Let's use the best forensic tool known to mankind, *Google*, and see what we can find.

"use block-list knowledge to perform special operations on files" finds http://old.lwn.net/2000/0413/announce.php3, a Linux information source, which refers to a Linux program called bmap as follows:

bmap 1.0.16 Use block-list knowledge to perform special operations on files.

Unfortunately the link to bmap is broken on this website.

Let's try **"Keld Simonsen".** Well, Keld is a popular guy. He's all over the Internet, on web pages, and newsgroups. In addition to the email address above there's another Keld.Simonsen@dkuug.dk. The domain, dkuug.dk is registered in Demark. This search is just leading me in circles so I'll go back to the previous hit.

“bmap” by itself yields too many hits, I’ll try **“Lunix bmap”,** about halfway down the first page I see the page http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html titled *Linux Data Hiding and Recovery*. This article discusses techniques for hiding data in file slack space.

Here’s a section of that article describing the functions of the bmap program.

```
The obscure tool bmap exists to jam data in slack space, take it
out and also wipe the slack space, if needed. Some of the examples
follow:

# echo "evil data is here" | bmap --mode putslack /etc/passwd

puts the data in slack space produced by /etc/passwd file

# bmap --mode slack /etc/passwd
getting from block 887048
file size was: 9428
slack size: 2860
block size: 4096
evil data is here

shows the data:

# bmap --mode wipeslack /etc/passwd

cleans the slack space.
```

The article links to ftp://ftp.scyld.com/pub/forensic_computing/bmap/ where I see bmap-1.0.20.tar.gz. Is this version “1.0.20” like we saw near the July 15th date in the file?

I downloaded the bmap-1.0.20.tar.gz file. There is apparently C programming source Inside the GZIPed file. Let’s see if our text strings are in there. I performed a simple Windows search of the files containing specified text

The strings

> **“use block-list knowledge to perform special operations on files”**
> **“display fragmentation information for the file”**
> **“wipe the file from the raw device”**

are in the bmap.c file.

Let's look at the bmap.c file contents. Here's a fragment of the file:

```
static struct mft_option options[]={
      {"doc","autogenerate document ...",
            MOT_VENUM|MOF_SILENT,
            MO_VENUM_CAST{
                  {"version","display version and exit",
                        0,MO_INT_CAST(BMAP_VERSION)
                  },
                  {"help","display options and exit",
                        0,MO_INT_CAST(BMAP_HELP)
                  },
                  {"man","generate man page and exit",
                        MOF_HIDDEN,MO_INT_CAST(BMAP_MAN)
                  },
                  {"sgml","generate SGML invocation info",
                        MOF_HIDDEN,MO_INT_CAST(BMAP_SGML)
                  },
                  {NULL,NULL,0,MO_CAST(NULL)}
            }
      },
      {"mode","operation to perform on files",
            MOT_VENUM|MOF_SILENT,
            MO_VENUM_CAST{
                  {"map","list sector numbers",
                        0,MO_INT_CAST(BMAP_MAP)},
                  {"carve","extract a copy from the raw device",
                        0,MO_INT_CAST(BMAP_CARVE)},
                  {"slack","display data in slack space",
                        0,MO_INT_CAST(BMAP_SLACK)},
                  {"putslack","place data into slack",
                        0,MO_INT_CAST(BMAP_PUTSLACK)},
                  {"wipeslack","wipe slack",
                        0,MO_INT_CAST(BMAP_WIPESLACK)},
                  {"checkslack","test for slack (returns 0 if file has
slack)",
                        0,MO_INT_CAST(BMAP_CHECKSLACK)},
                  {"slackbytes","print number of slack bytes
available",0,MO_INT_CAST(BMAP_SLACKBYTES)},
                  {"wipe","wipe the file from the raw
device",0,MO_INT_CAST(BMAP_WIPE)},
                  {"frag","display fragmentation information for the
file",0,MO_INT_CAST(BMAP_FRAGMENT)},
                  {"checkfrag","test for fragmentation (returns 0 if file
is fragmented)",0,MO_INT_CAST(BMAP_CHECKFRAG)},
                  {NULL,NULL,0,MO_CAST(NULL)}}
```

It appears that this section of code determines the mode of operation such as "place data into slack" or "wipe slack".

Since the selected text strings from the prog file are found in the bmap source I'm thinking that prog is simply a renamed version of the bmap program.

Since the executable ELF file is statically linked, it has libraries embedded. The corresponding libraries on my operating system are most likely different than the ones used to compile to file in question. The chance that suspects file and a freshly complied file on my system having a matching MD5 is virtually zero. So let's compare the operation of the suspect file versus the Bmap program to see if they are the same.

Running an unknown binary file can severely affect the operation and stability of the system. At this point we really don't know if we have malicious code or not. I'll run the binary on our forensic PC, I can easily recreate the operating system from an image that I have just for this purpose. Also, the forensic PC is not attached to another computer or the Internet and there are no other storage devices physically attached.

One of the main issues is to determine if the program attempts to spawn a process or it simply executes and terminates.

By checking the running processes before and after executing the binary we can see if it has spawned any processes. The ps command reports the status of running processes.

The following is a side-by-side comparison of the output of the ps command, with –A option which shows ALL processes.

BEFORE

PID	TTY	TIME	CMD
1	?	0:00:05	init
2	?	0:00:00	migration/0
3	?	0:00:00	migration/1
4	?	0:00:00	keventd
5	?	0:00:00	ksoftirqd_CPU0
6	?	0:00:00	ksoftirqd_CPU1
11	?	0:00:00	bdflush
7	?	0:00:00	kswapd
8	?	0:00:00	kscand/DMA
9	?	0:00:03	kscand/Normal
10	?	0:00:00	kscand/HighMem
12	?	0:00:00	kupdated
13	?	0:00:00	mdrecoveryd
17	?	0:00:00	kjournald
75	?	0:00:00	khubd
3595	?	0:00:00	kjournald
3649	?	0:00:00	knodemgrd
3966	?	0:00:00	dhclient
4017	?	0:00:00	syslogd
4021	?	0:00:00	klogd
4039	?	0:00:00	portmap
4058	?	0:00:00	rpc.statd
4154	?	0:00:00	sshd
4168	?	0:00:00	xinetd
4177	?	0:00:00	gpm
4186	?	0:00:00	crond
4257	?	0:00:00	xfs
4275	?	0:00:00	atd
4285	?	0:00:00	rhnsd
4291	tty1	0:00:00	mingetty
4292	tty2	0:00:00	mingetty
4293	tty3	0:00:00	mingetty
4294	tty4	0:00:00	mingetty

AFTER

PID	TTY	TIME	CMD
1	?	0:00:05	Init
2	?	0:00:00	migration/0
3	?	0:00:00	migration/1
4	?	0:00:00	keventd
5	?	0:00:00	ksoftirqd_CPU0
6	?	0:00:00	ksoftirqd_CPU1
11	?	0:00:00	bdflush
7	?	0:00:00	kswapd
8	?	0:00:00	kscand/DMA
9	?	0:00:03	kscand/Normal
10	?	0:00:00	kscand/HighMem
12	?	0:00:00	kupdated
13	?	0:00:00	mdrecoveryd
17	?	0:00:00	kjournald
75	?	0:00:00	khubd
3595	?	0:00:00	kjournald
3649	?	0:00:00	knodemgrd
3966	?	0:00:00	dhclient
4017	?	0:00:00	syslogd
4021	?	0:00:00	klogd
4039	?	0:00:00	portmap
4058	?	0:00:00	rpc.statd
4154	?	0:00:00	sshd
4168	?	0:00:00	xinetd
4177	?	0:00:00	gpm
4186	?	0:00:00	crond
4257	?	0:00:00	xfs
4275	?	0:00:00	atd
4285	?	0:00:00	rhnsd
4291	tty1	0:00:00	mingetty
4292	tty2	0:00:00	mingetty
4293	tty3	0:00:00	mingetty
4294	tty4	0:00:00	mingetty

4295	tty5	0:00:00	mingetty
4296	tty6	0:00:00	mingetty
4297	?	0:00:00	gdm-binary
4340	?	0:00:00	gdm-binary
4341	?	0:00:52	X
4449	?	0:00:00	cupsd
14613	?	0:00:00	gnome-session
14671	?	0:00:00	ssh-agent
14682	?	0:00:00	gconfd-2
14684	?	0:00:00	bonobo-activati
14686	?	0:00:00	gnome-settings-
14691	?	0:00:00	fam
14698	?	0:00:08	metacity
14702	?	0:00:02	gnome-panel
14704	?	0:00:12	nautilus
14706	?	0:00:02	magicdev
14708	?	0:00:00	eggcups
14710	?	0:00:00	pam-panel-icon
14712	?	0:00:24	rhn-applet-gui
14713	?	0:00:00	pam_timestamp_c
14720	?	0:00:00	notification-ar
14771	?	0:00:03	gnome-terminal
14772	?	0:00:00	gnome-pty-helpe
14773	pts/0	0:00:00	bash
14852	?	0:00:00	mapping-daemon
15181	pts/0	0:00:00	ps

4295	tty5	0:00:00	mingetty
4296	tty6	0:00:00	mingetty
4297	?	0:00:00	gdm-binary
4340	?	0:00:00	gdm-binary
4341	?	0:00:53	X
4449	?	0:00:00	cupsd
14613	?	0:00:00	gnome-session
14671	?	0:00:00	ssh-agent
14682	?	0:00:00	gconfd-2
14684	?	0:00:00	bonobo-activati
14686	?	0:00:00	gnome-settings-
14691	?	0:00:00	fam
14698	?	0:00:08	metacity
14702	?	0:00:02	gnome-panel
14704	?	0:00:13	nautilus
14706	?	0:00:02	magicdev
14708	?	0:00:00	eggcups
14710	?	0:00:00	pam-panel-icon
14712	?	0:00:24	rhn-applet-gui
14713	?	0:00:00	pam_timestamp_c
14720	?	0:00:00	notification-ar
14771	?	0:00:03	gnome-terminal
14772	?	0:00:00	gnome-pty-helpe
14773	pts/0	0:00:00	bash
14852	?	0:00:00	mapping-daemon
15188	pts/0	0:00:00	ps

We can see that there are no new processes between the before and after ps –A output listings. The only difference is the last entry, which is the ps command itself. From this we can conclude that the program does not create any new processes.

The strace command will trace system calls on an executable.
The command strace –fxi ./prog yields the following output:

> The -f option traces child processes, -i prints the instruction pointer and –x prints non-ASCII in hex format.

```
14941 [080480e0] execve("./prog", ["./prog"], [/* 33 vars */]) = 0
14941 [0805531e] fcntl64(0, F_GETFD)    = 0
14941 [0805531e] fcntl64(1, F_GETFD)    = 0
14941 [0805531e] fcntl64(2, F_GETFD)    = 0
14941 [0806f9ed] uname({sys="Linux", node="localhost.localdomain", ...}) = 0
14941 [0806fad0] geteuid32()            = 0
14941 [0806fa64] getuid32()             = 0
14941 [0806fba8] getegid32()            = 0
14941 [0806fb3c] getgid32()             = 0
14941 [080700b5] brk(0)                 = 0x80bedec
14941 [080700b5] brk(0x80bee0c)         = 0x80bee0c
14941 [080700b5] brk(0x80bf000)         = 0x80bf000
14941 [080700b5] brk(0x80c0000)         = 0x80c0000
14941 [08055114] write(2, "no filename. try \'--help\' for he"..., 36) = 36
14941 [08054ced] _exit(2)               = ?
```

We do not see any fork system calls, which again implies no new processes launched by the program. The program exits after writing "no filename. try '—help' for help" to the display. Apparently the program is asking for a file name to work on.

Analysis of ps and strace command outputs indicate that the binary does not spawn a new process and operates on a specified file. This supports the suspicion that the binary is a renamed version of Bmap. We can now execute the binary and check its results.

Simply executing the binary with no command line parameters yields the following message on the display:
`“no filename. try ‘—help’ for help.”`
This is consistent with the strace output and the request for a file name to work on.

Next let’s look at the included help by executing prog –help. Following is the output:

```
prog:1.0.20 (07/15/03) newt
Usage: prog [OPTION]... [<target-filename>]
use block-list knowledge to perform special operations on files

--doc VALUE
  where VALUE is one of:
  version  display version and exit
  help  display options and exit
  man  generate man page and exit
  sgml  generate SGML invocation info
--mode VALUE
  where VALUE is one of:
  m  list sector numbers
  c  extract a copy from the raw device
  s  display data
  p  place data
  w  wipe
  chk  test (returns 0 if exist)
  sb  print number of bytes available
  wipe  wipe the file from the raw device
  frag  display fragmentation information for the file
  checkfrag  test for fragmentation (returns 0 if file is fragmented)
--outfile <filename> write output to ...
--label useless bogus option
--name  useless bogus option
--verbose      be verbose
--log-thresh <none | fatal | error | info | branch | progress | entryexit> logging threshold ...
--target <filename> operate on ...
```

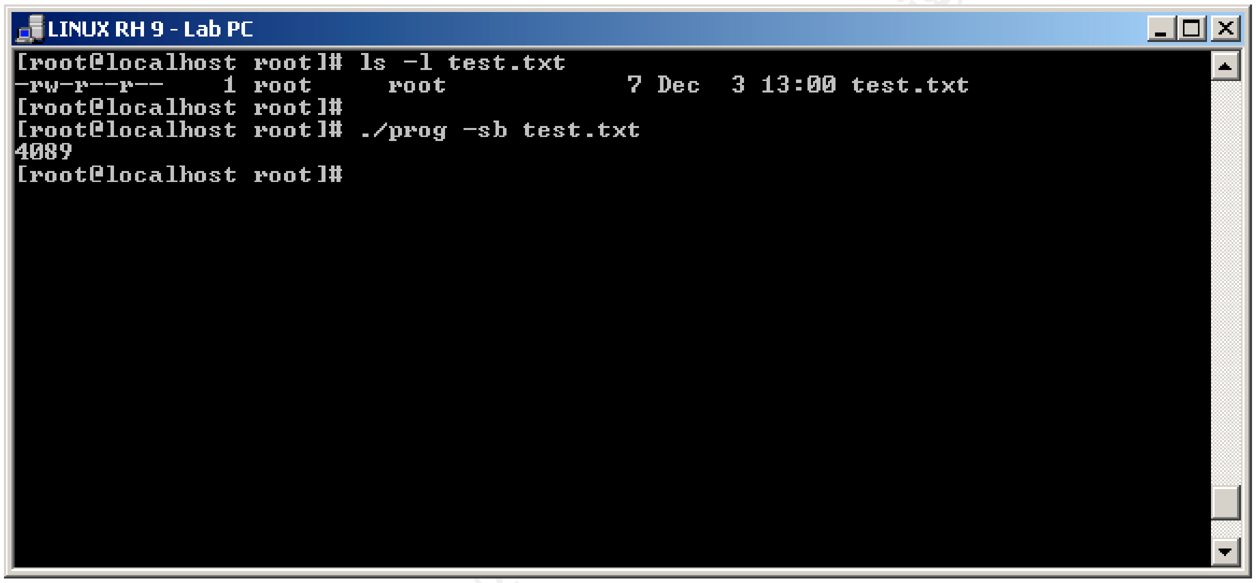
The output created by executing the downloaded version of Bmap yields almost identical results. The only differences are in the –mode option values. In the suspect version of the executable some of the mode options are abbreviated such as “m” instead of “map” and “c” instead of “carve” and so on. These minor cosmetic changes to the help listing could have easily been made at compile time.

Now that we know the binary is used to store data in the slack space of files, let discuss slack space. A computer operating system allocates disk space in “chunks” called sectors. Often, a file does not fit exactly into an even number of sectors. This creates “wasted” space between the end of the file and the end of its last allocated sector. The wasted space is called slack. The slack space is not normally visible to typical application programs because these programs stop reading at the logical end of the file, ignoring the space remaining in the last allocated sector.

A program like Bmap (our unknown binary) could be used to secretly store data in the slack space of other files. When the files are viewed by their native application, the secret information would be invisible.

Let' test the operation by storing some information in the slack space of a file.

I'll create a small text document by issuing the command **`echo tester >test.txt.`** This creates a file that is 7 bytes long (the specified text plus an end-of-file byte). The command **`ls –l test.txt`** confirms the length at 7 bytes. Running the –sb mode (show slackbytes) of the prog binary shows that there are 4089 bytes of slack space at the end of the test.txt file.

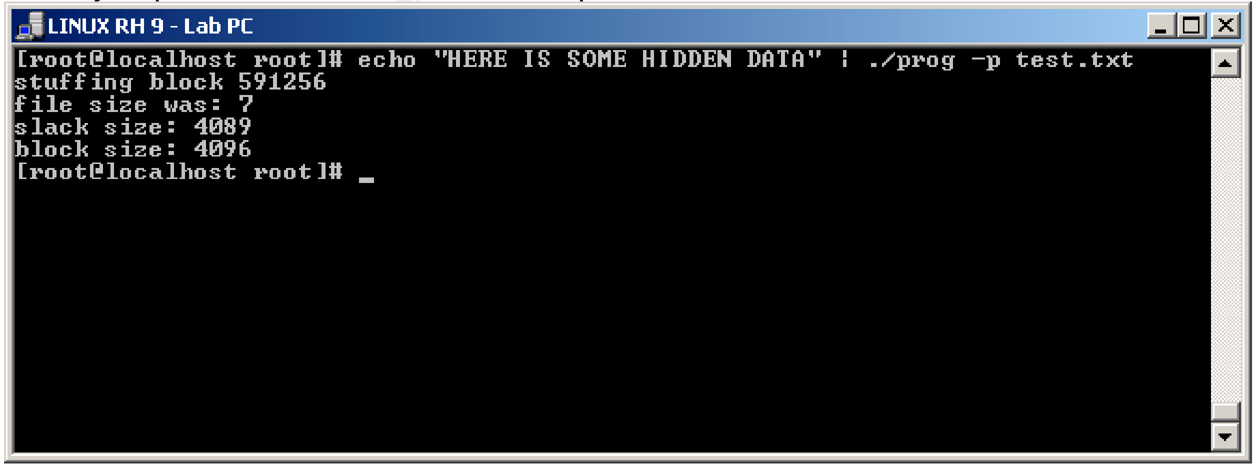


LINUX RH 9 - Lab PC

```
[root@localhost root]# ls -l test.txt
-rw-r--r--    1 root     root            7 Dec  3 13:00 test.txt
[root@localhost root]#
[root@localhost root]# ./prog -sb test.txt
4089
[root@localhost root]#
```

This would make sense because the Linux file system in use allocates sectors with 4096 bytes each.

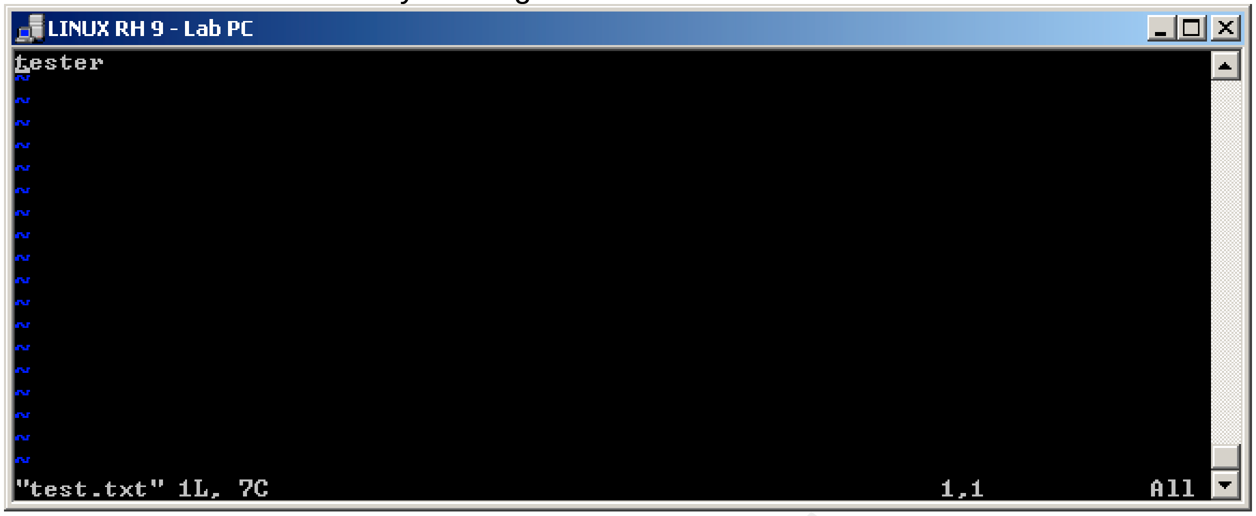
Lets try to put some data into the slack space.



LINUX RH 9 - Lab PC

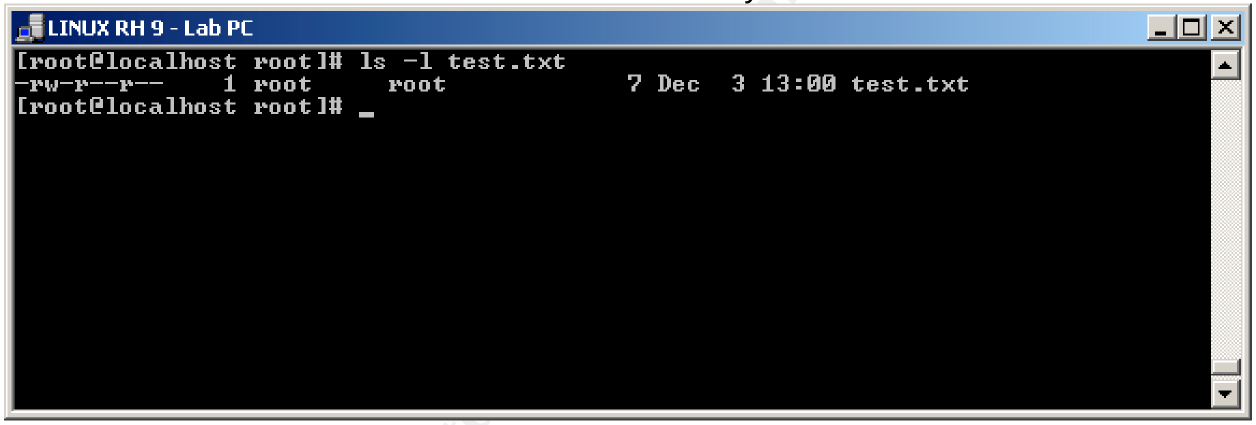
```
[root@localhost root]# echo "HERE IS SOME HIDDEN DATA" | ./prog -p test.txt
stuffing block 591256
file size was: 7
slack size: 4089
block size: 4096
[root@localhost root]# _
```

The VI text editor shows only the original contents of the file without the hidden text.



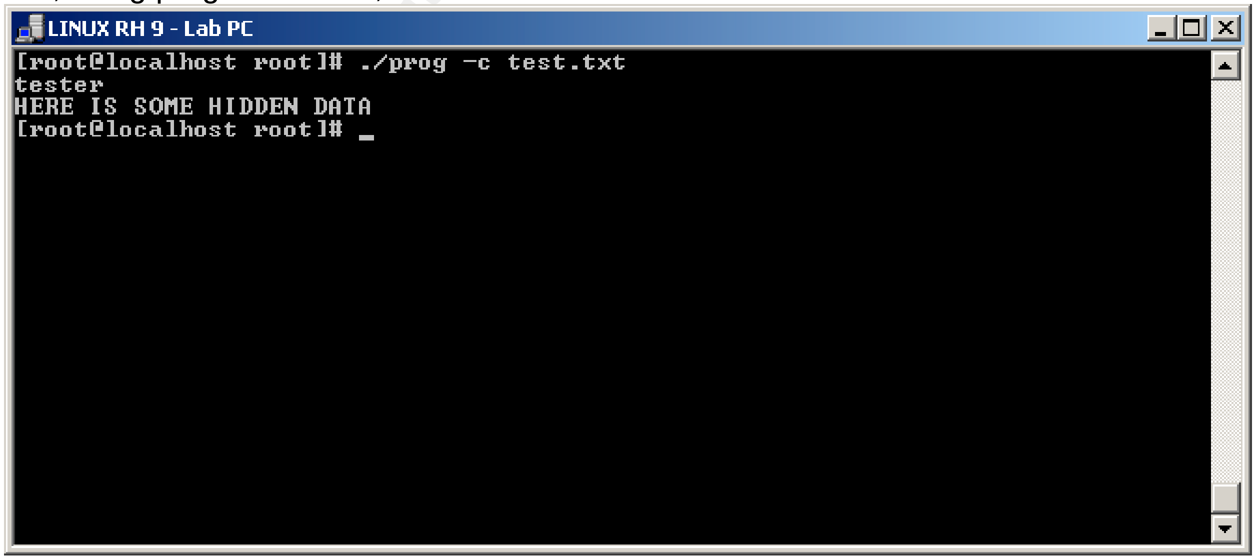
```
tester
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"test.txt" 1L, 7C                                                    1,1           All
```

…and the ls command still shows the file size as 7 bytes



```
[root@localhost root]# ls -l test.txt
-rw-r--r--    1 root     root            7 Dec  3 13:00 test.txt
[root@localhost root]# _
```

Yet, using prog's –c mode, we can see the hidden data.



```
[root@localhost root]# ./prog -c test.txt
tester
HERE IS SOME HIDDEN DATA
[root@localhost root]# _
```

By running similar tests with my downloaded version of Bmap, I get the same response. As a matter of fact I can interchangeable use the prog binary and Bmap program to achieve the same results.

At this point I can conclude that the prog binary file is functionally the same as the Bmap program. These programs are designed to test files for slack space and insert data in file slack space where it will be hidden from typical users.

Forensic Details – Since the ELF binary was statically linked it was most likely complied on the suspect's system or one with a similar operating system and libraries. If we had access to the operating system we might see source code or bash history.

This operating system captures 3 times associated with the prog file. These times are referred to as MAC times. "M" modification – is the time the file was last modified. "A" accessed – is the time the file was last accessed. "C" changed – is the time the file was created or last changed. The stat command displays these times.

The last access time was 02:12:45 on July, 16 2003 which can indicate the last time the program was executed. Assuming of course if the computer system's time clock was set correctly along with the time zone. Also, any other program that accesses the files on the diskette such as anti virus programs can affect the last accessed time.

The program does not alter the operation of any programs or data; it just tacks on hidden data in the slack space at the logical end of a file. Again, any data inserted into the slack area of a program file would not affect the operation of that program. The chance that this program would affect the operation of an operating system is unlikely.

Legal Implications – If a program is not executed since its creation, the MAC times would all be the same (Modified, Accessed and Changed all at the same moment in time). When a file is copied or moved to a floppy drive just the Accessed and Changed times are updated and the Modified time remains. In our case we see the Modified time as 10:24:00 on July, 14 2003 and both the Accessed and Changed times 2 calendar days later with the last accessed time about 7 minutes later. While not proof, these times are consistent with the user creating/compiling the file on Monday, July 14th, copying the file to floppy on Wednesday July 16th and last executing it about 7 minutes later. We have no traces of how often it was executed. Again, there are other possible explanations for the last access time, so we don't have proof the file was run. It would be interesting to check timecards and work logs to see Mr. Price's work hours on Tuesday July 15th 2003.

Since we cannot prove Mr. Price ran the suspect program, we can't determine any malicious activity other than possibly violating corporate policy against unauthorized software.

Also, the program serves no other purpose other than to check for slack space and hide data in the slack space. What business application purpose could this serve?

Mr. Price purposely wiped his hard drive before the Incident Response Team arrived in an apparent attempt to cover up his activities. It can be argued that this is not the act of an innocent man. Additionally, his actions could be considered sabotage and destructive of company property requiring significant resources to reload and configure the PC for use by another employee.

If Mr. Price was distributing copyrighted material using our company computers, we may face legal liability. We could be accused of not being responsible for managing Internet access and employee activities. Considerable company resources could be expended simply defending ourselves in legal proceedings.

Interview Questions – In our scenario Mr. Price has already been suspended from employment. Obviously he knows he's be accused so a subsequent interview might be confrontational.

Many conflicts are quickly terminated once the suspect is faced with the results of a forensic analysis.

> 1. "Mr. Price, a forensics analysis of your computer and other media has revealed significant information about your recent activities. Do you want to tell us what unauthorized activities you've been using your computer for?"

We need to establish his presence at work on the days in question.

> 2. "What was your work schedule during the week of July 14, 2003? Specifically, did you work late on Tuesday July 15th?"

Let's determine his compliance with policy and establish other's access to his computer.

> 3. "Did you adhere to company policy and change your password every 30 days? When was the last time you changed your password? Does anyone else have your password? Did you login as administrator to you local machine?"

Verify his understanding of policy regarding use of unauthorized programs and copyrighted material.

> 4. "Our company policy states that there shall be no use of unauthorized software or software for non-business use without prior approval of the IT department and

your immediate supervisor. Were there any unauthorized programs on your computer?"

5. "Do you understand that the unauthorized redistribution of any copyrighted material including MP3 audio files and DVD movies is a violation of the law?"

Let's try again.

6. "Were your engaged in any unauthorized or illegal activity using your office computer?"

Now we'll call him on specific findings.

7. "Graphic files were found depicting hard drive geometry and sector layout. How did these files support your business activities? What did you use them for?"

8. "A copy of NetCat was found. The only purpose of this program is to read and write data across the network? Who's data were you accessing and why?"

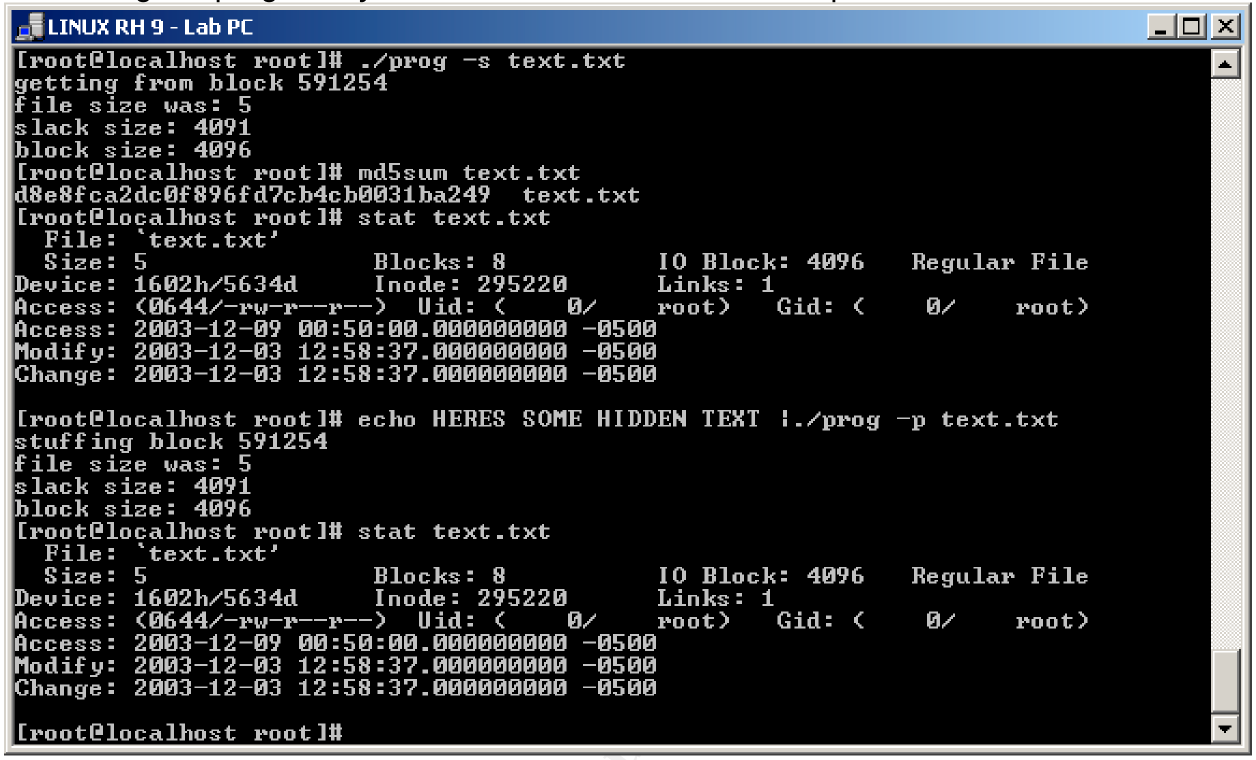
9. "When you wrote the message to Mike stating `I received the latest batch of files last night and I'm ready to rock-n-roll (ha-ha). I have some advance orders for the next run. Call me soon. JP` What did you mean? What orders were you speaking of?"

10. "Why did you need information on recompiling the Linux Kernel, playing sound MP3's and DVD's in Linux and how are these business related?"

11. "What were you trying to hide when you wiped the hard drive of your computer? A simple format would have deleted information and it would have been much faster."

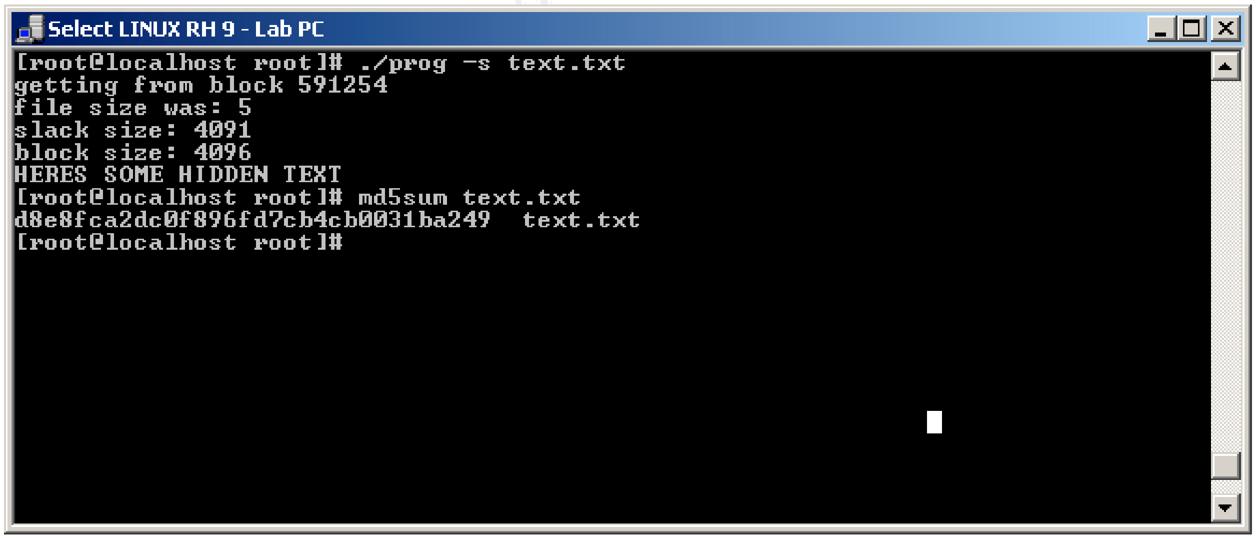
Case Information – The use of this binary program is very difficult to detect. MAC times of the affected files are not changed nor are the MD5 hashes changed.

To illustrate this we'll run an MD5 hash on a test file and a stat to get it's MAC times, then using the prog binary, add some hidden text and compare the hash and times.



```
[root@localhost root]# ./prog -s text.txt
getting from block 591254
file size was: 5
slack size: 4091
block size: 4096
[root@localhost root]# md5sum text.txt
d8e8fca2dc0f896fd7cb4cb0031ba249  text.txt
[root@localhost root]# stat text.txt
  File: `text.txt'
  Size: 5               Blocks: 8          IO Block: 4096   Regular File
Device: 1602h/5634d     Inode: 295220      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2003-12-09 00:50:00.000000000 -0500
Modify: 2003-12-03 12:58:37.000000000 -0500
Change: 2003-12-03 12:58:37.000000000 -0500

[root@localhost root]# echo HERES SOME HIDDEN TEXT |./prog -p text.txt
stuffing block 591254
file size was: 5
slack size: 4091
block size: 4096
[root@localhost root]# stat text.txt
  File: `text.txt'
  Size: 5               Blocks: 8          IO Block: 4096   Regular File
Device: 1602h/5634d     Inode: 295220      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2003-12-09 00:50:00.000000000 -0500
Modify: 2003-12-03 12:58:37.000000000 -0500
Change: 2003-12-03 12:58:37.000000000 -0500

[root@localhost root]#
```

Notice that none of the MAC times changed even though I added some data to the slack space.

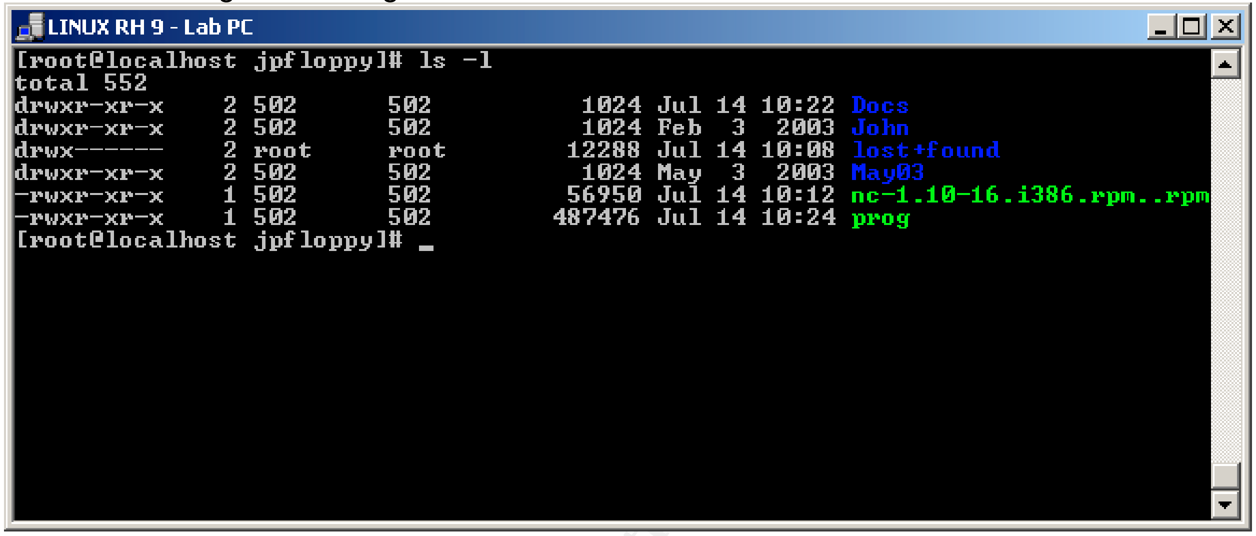


```
[root@localhost root]# ./prog -s text.txt
getting from block 591254
file size was: 5
slack size: 4091
block size: 4096
HERES SOME HIDDEN TEXT
[root@localhost root]# md5sum text.txt
d8e8fca2dc0f896fd7cb4cb0031ba249  text.txt
[root@localhost root]#
```

Also, the MD5 hash did not change.

To determine if the program was used, a script could be written to run Bmap on all files located on a suspect system using the –c option which extracts a copy of the hidden data, then accumulate the output in a text file for later analysis.

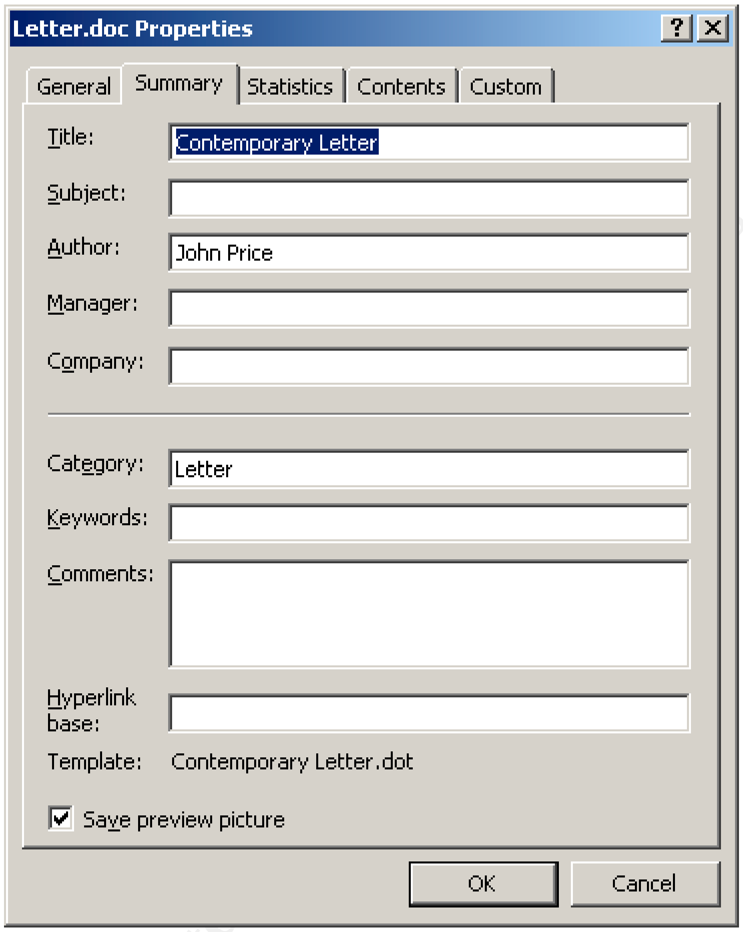
A quick ls of the mounted floppy drive image shows a file nc-1.10-16.i386.rpm..rpm. A Google search indicates that this is an installation package for NetCat. The utility is used for reading and writing data over and TCP or UDP network.



```
[root@localhost jpfloppy]# ls -l
total 552
drwxr-xr-x    2 502      502          1024 Jul 14 10:22 Docs
drwxr-xr-x    2 502      502          1024 Feb  3  2003 John
drwx------    2 root     root        12288 Jul 14 10:08 lost+found
drwxr-xr-x    2 502      502          1024 May  3  2003 May03
-rwxr-xr-x    1 502      502         56950 Jul 14 10:12 nc-1.10-16.i386.rpm..rpm
-rwxr-xr-x    1 502      502        487476 Jul 14 10:24 prog
[root@localhost jpfloppy]# _
```

Using prog along with NetCat the user could easily hide data in the slack space of files across the network.

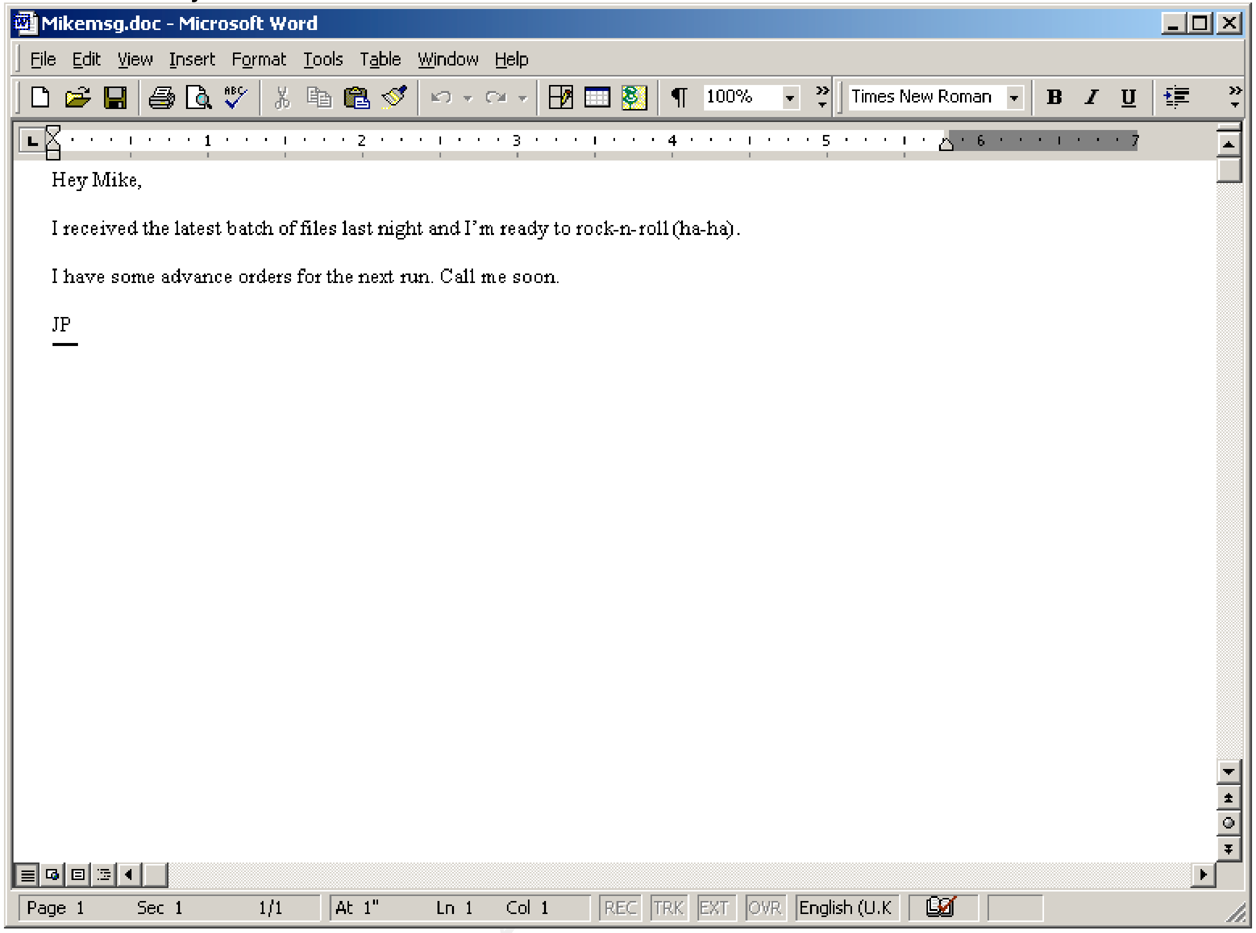
Looking in the Docs directory we find Letter.doc and Mikemsg.doc both Microsoft Word documents. Letter.doc is simply a Word "Contemporary Letter" template. Word metadata contains "John Price" as the author.



Title:	Contemporary Letter
Subject:	
Author:	John Price
Manager:	
Company:	
Category:	Letter
Keywords:	
Comments:	
Hyperlink base:	
Template:	Contemporary Letter.dot

[✓] Save preview picture

The Mikemsg.doc file contains the text of a message probably from Mr. Price to unknown subject named Mike.


Mikemsg.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
100%
Times New Roman
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 REC TRK EXT OVR English (U.K

Hey Mike,

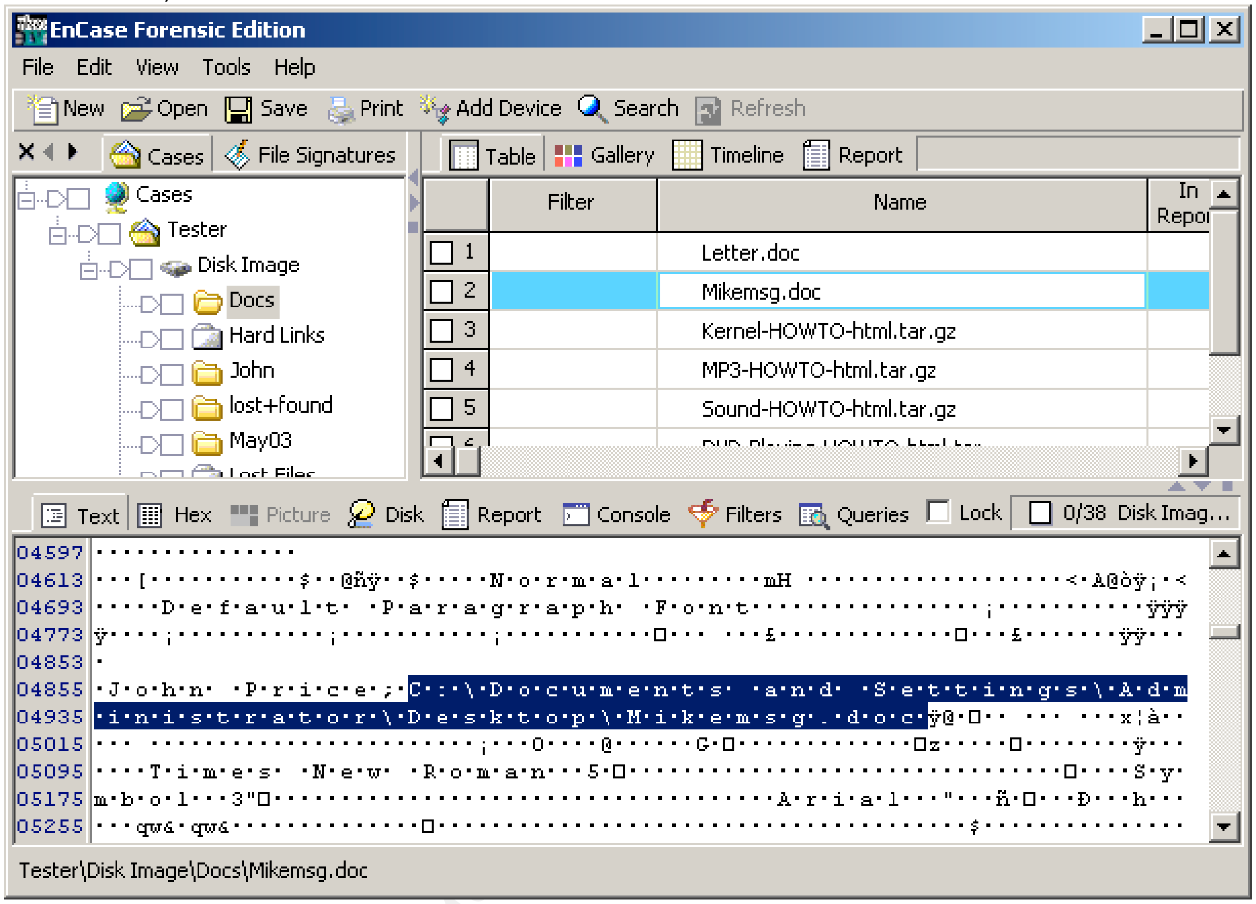
I received the latest batch of files last night and I'm ready to rock-n-roll (ha-ha).

I have some advance orders for the next run. Call me soon.

JP

While we don't have enough context to be assured, we can get the feeling of some covert activity. The phrase **`"latest batch of files"`** could mean ongoing activity. **`"I'm ready to rock-n-roll(ha-ha)"`** may be a play on words indicating not only is he ready, but music files are involved. There is also indication of more activity in the future by the phrase **`"advance orders".`** By using the term **`"orders"`** are they collecting money for their activities? If Napster got in trouble for facilitating free file distribution, imagine if they charged for the service, like these guys.

If we look at the raw text contained in the Word document file, we see some Word metadata indicating that the original file was located on the administrator's desktop. This is evidence that the file was once located on Administrator user's desktop. Remember, Mr. Price denied that the diskette was his.


EnCase Forensic Edition
File Edit View Tools Help
New Open Save Print Add Device Search Refresh
Cases File Signatures
Cases
Tester
Disk Image
Docs
Hard Links
John
lost+found
May03
Lost Files
Table Gallery Timeline Report
Text Hex Picture Disk Report Console Filters Queries Lock 0/38 Disk Imag...

	Filter	Name	In Repor
1		Letter.doc	
2		Mikemsg.doc	
3		Kernel-HOWTO-html.tar.gz	
4		MP3-HOWTO-html.tar.gz	
5		Sound-HOWTO-html.tar.gz	

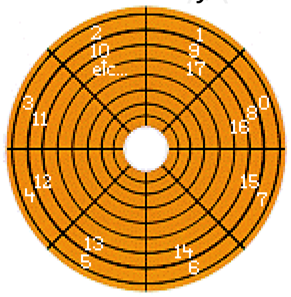
```
04597 ················
04613 ···[···········$··@ñÿ··$·····N·o·r·m·a·l·········mH ···················<·A@òÿ¡·<
04693 ·····D·e·f·a·u·l·t· ·P·a·r·a·g·r·a·p·h· ·F·o·n·t·················¡···········ÿÿÿ
04773 ÿ····¡············¡············¡············□··· ···£·············□···£·······ÿÿ···
04853 ·
04855 ·J·o·h·n· ·P·r·i·c·e·;·C·:·\·D·o·c·u·m·e·n·t·s· ·a·n·d· ·S·e·t·t·i·n·g·s·\·A·d·m
04935 ·i·n·i·s·t·r·a·t·o·r·\·D·e·s·k·t·o·p·\·M·i·k·e·m·s·g·.·d·o·cÿ@·□·· ··· ···x¦à··
05015 ··· ·························¡···O····@······G·□·············□z·····□········ÿ···
05095 ····T·i·m·e·s· ·N·e·w· ·R·o·m·a·n···5·□·····························□····S·y·
05175 m·b·o·l···3"□·····························A·r·i·a·l···"···ñ·□···Ð···h···
05255 ···qw&·qw&··············□·····························$··············
```

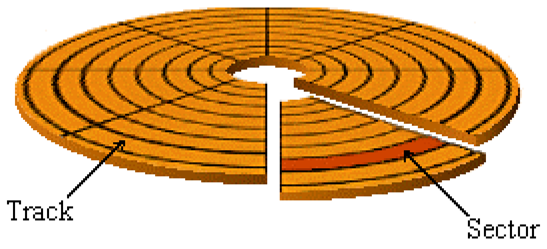
Tester\Disk Image\Docs\Mikemsg.doc

This is in the folder structure for Windows 2000 or Windows XP operating system.

Also in this folder are several GZIPed files. They are tar archive files of HTML documents describing How to configure and compile the Linux kernel, How to configure Linux for sound, How to Play MP3 audio files on Linux and How to play DVD movies on Linux.

In the directory labeled John, there are 2 gif image files sect-num.gif and sectors.gif





Both of these files appear to illustrate basic drive geometry. This would support user research for using the Bmap file slack utility program.

Form the limited information on John Price’s floppy disk it’s difficult to say with certainty that he was distributing copyrighted material – we do not have the “smoking gun”. He definitely had some obscure, non-business related tools and information which probably violated company policy.

Like most cases involving computer forensics, electronic evidence is only one price of the puzzle. The interview with Mr. Price could be very revealing along with information gathered from the other suspect, Mike.

Part 2 -- Perform Forensic Tool Validation

Background and Scope -- One of the most important functions of a sound computer forensic analysis is data acquisition, specifically hard drive acquisitions in the field. Many times in the corporate environment it's not practical to "seize" the suspect computer and bring it back to the lab for acquisition. Also, many investigations include "black bag" acquisitions, those done covertly, without the user's knowledge. My experience in the corporate world is that I need to get in and get out quickly. Law enforcement agents may come across situations were consent is given to search a computer but the suspect will not allow the computer to be removed.

I will be validating a handheld disk-duplicating device designed specifically for computer forensics application. For this tool to be accepted it must duplicate the source drive exactly bit-for-bit and it must not change any data on the suspect's hard drive. I will use other methods of disk imaging then compare the results to the handheld unit thus verifying its performance.

Let's talk for a moment about drive acquisition: for a disk copy to be forensically sound the entire contents of the drive must be duplicated exactly bit-for-bit. This type of copying has been termed "bit-stream" copying. Disk cloning utilities used by IT staff for backup and baseline deployment of new PCs may or may not perform a true bit-stream copy. For example, to increase performance and reduce disk image storage space, some utilities do not copy unused disk space. If I were to use a copy of a hard drive that did not contain unused disk space I could not recover deleted files.

In the world of forensics we want to perform analysis on the "best evidence" which by definition would be the original hard drive. However; due to the fragile nature of electronic evidence, a copy of the evidence is acceptable but only if that copy can be demonstrated to be authentic.

Tool Description -- There are a couple of approaches to field acquisitions. One is to use a laptop or portable computer system with a write-inhibitor device and physically attach the suspect's hard drive and create a bit-stream copy using software designed for that purpose. The other technique is to use a portable hard drive duplicator, which makes the bit-stream copy to a secondary hard drive mounted inside the unit. I will be validating the former, a portable, handheld IDE hard drive disk duplicator.

My tests will be on the "Forensics SF-5000u™" device manufactured by Logicube located in Chatsworth, CA. Logicube's website is at www.logicube.com.

The unit is constructed of bright yellow plastic and is 5½" wide, 8" long and about 2½" deep, it weighs about one pound without a hard drive installed.   The unit is hinged at the top and opens in a clamshell fashion allowing the installation of a standard IDE hard drive inside. An LCD display is located on the front of the unit along with several buttons that control the operation of the unit. Power is supplied using an external AC power adapter. The top of the unit has 3 external connections; standard male 40-pin IDE for connecting the suspect's hard drive, a 25-pin female connector for a parallel printer and a 4-pin power adapter for supplying DC power to the suspect hard drive.

The SF-5000u ships from Logicube as a kit containing:

- Rugged carrying case, with power distribution panel
- A portable printer
- 5", 9" and 18" IDE ribbon cables
- A 2½" drive adapter for laptop hard drives
- Logicube CloneCard Pro™, a PCMCIA adapter card for laptop acquisitions
- 25-pin printer cable
- Flashlight and screw driver

The SF-5000u can also be purchased as a stand-alone unit, with a nylon case, IDE cables, flashlight and screwdriver

Basic Operation: A destination drive is mounted inside the unit; the source drive remains outside connected via a 40-pin IDE ribbon cable and separate power adapter. Using the front-panel buttons and LCD display, the user selects the mode of operation then the device, using built-in circuitry, performs a bit-for-bit, exact image of the source hard drive onto the destination drive.

The unit used in this test has firmware v2.0 and software v1.15. Logicube provides software to upload new firmware to the unit as needed.

The major advantages of using the Logicube SF-5000u are ease of use, primarily because of its portability, lack of moving parts (reliability), and cost as compared to a laptop with acquisition software and peripherals. The SF-5000u is operating system independent because it's simply a drive duplicator.

Test Apparatus -- I will be using an Intel-based workstation as a lab PC for verification of data capture. The motherboard is an Intel D865PERL (BIOS P08) with an Intel Pentium-4 2.6GHz CPU (stepping 8) and 512MB of DDR-400 memory.

Operating system is Linux® -- distribution is RedHat® 9, kernel 2.4.20-20.9

The lab PC will not be attached to a network nor will it connect to the Internet. The only storage devices connected during testing will be the 3½" floppy drive, the operating system drive and the suspect drive. The onboard IDE and floppy controllers will be used.

The destination hard drive placed inside the Logicube unit will be a new, Enhanced IDE Western Digital Caviar® 80.0Gb

```
        Model:                  WD800JB-OOCRA1
        Manufactured date:      10/19/03
        Rotational Speed:       7,200 RPM
        Buffer Size:            8 MB
        Geometry:               16,383 Cylinders
                                16 Heads (Logical)
                                63 Sectors Per Track
                                Resulting in 156,301,488 sectors
        Bytes Per Sector:       512
        Fastest Mode:           Mode 5 Ultra ATA - 100.0 MB/s

        Drive specifications taken from Western Digital website at
        http://www.westerndigital.com/en/products/products.asp?DriveID=32
```

The source or suspect hard drive is a used, Enhanced IDE Western Digital Caviar® 3.1Gb

```
        Model:                  AC33100-75H
        Manufactured date:      7/15/97
        Rotational Speed:       5,200 RPM
        Buffer Size:            128K
        Geometry:               6,136 Cylinders
                                16 Heads
                                63 Sectors Per Track
                                Resulting in 6,185,088 sectors
        Bytes Per Sector:       512
        Fastest Mode:           Mode 4 PIO - 16.6 MB/s

        Drive specifications taken from Western Digital website at
        http://support.wdc.com/productspec.asp (search for WDAC33100)
```

Procedures – I will capture the data using commonly accepted, forensically sound procedures to obtain a baseline signature of the data, and then use the SF-5000u to capture the data and compare the two. I will use before and after MD5 hashes to compare the data. MD5 is discussed on page 4. An outline of procedures follows:

1. Sanitize the destination drive
2. Get a baseline signature of suspect drive
3. Capture the suspect drive using the Logicube SF-5000u unit
4. Get a post-capture signature of suspect drive to ensure no changes were made to the drive during the capture process.
5. Verify that the destination drive copy exactly matches the suspect drive data

Criteria for Approval

1. All data from the suspect drive must be copied to the destination drive with no errors.

2. No data on the suspect drive may be altered in any way.

3. The device must provide feedback for success or failure.

4. Verification must be presented.

5. Copying speed must be roughly comparable to typical direct disk-to-disk method.

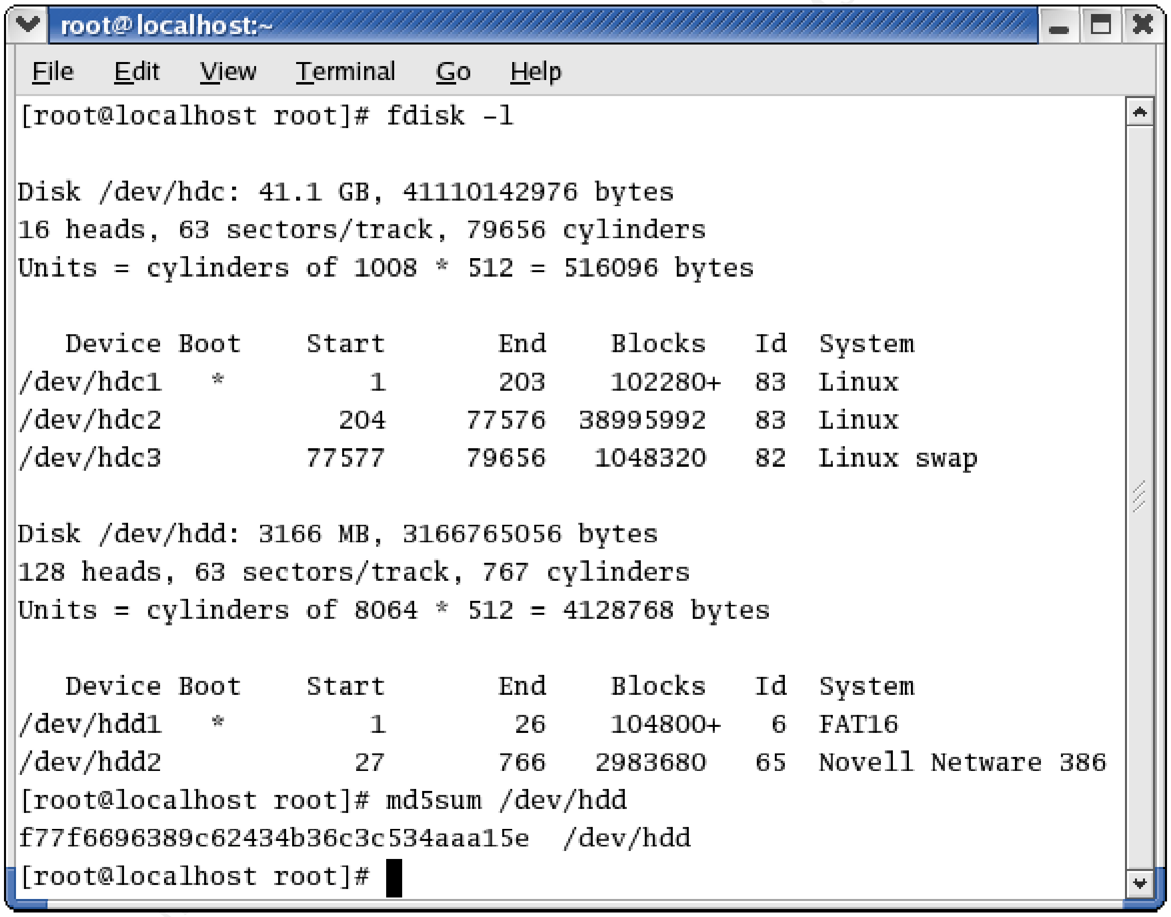
Data and Results

1. **Sanitize the destination drive –** We must start with a clean destination media. This prevents cross-contamination from other cases if the drive had previously been used. Even drives directly from the manufacturer have some garbage data on them. I will use Linux **`dd`** to write zeros to the entire hard drive.
   a. Using a removable IDE drive bay, I installed the destination drive in the lab PC jumpered as a slave drive. (Linux OS is on the master)
   b. Boot into Linux.
   c. Use **`fdisk –l`** to list the partition tables of the hard drives. I like to see all the partition tables together, it gives me a better feeling for what's going before I wipe a drive.
   d. Using **`dd`**, I will overwrite the destination drive with zero's by inputting the pseudo device /dev/zero to the hard drive

```
# dd  if=/dev/zero of=/dev/hdd
```

```
It's important to note that the SF-5000u has a wipe drive
function built-in.  When its drive wipe is complete, the
SF-5000 writes a small signature on the drive indicating
the wipe was performed.  This allows the SF-5000u to detect
that a previously wiped drive is loaded inside the unit;
therefore it will not ask to wipe the drive during the
capture process.

Optionally, the signature writing procedure can be disabled
when wiping a drive.  Also, if a signature is not present
at capture time you can override and proceed with the
capture anyway.

For this demonstration I have elected to use a dd disk
wiping process of the destination drive.

An examination of the SF-5000u WipeClean function is in
Appendix A at the end of the document.
```

2. **Get a baseline signature of suspect drive –** I will obtain an MD5 hash of the suspect drive.
   a. Using a removable IDE drive bay, I installed the suspect drive in the lab PC jumpered as a slave drive.
   b. Boot into Linux.
   c. Use **`fdisk -l`** to list the partition tables of the hard drives.
   d. Run **`MD5SUM`** against the suspect drive

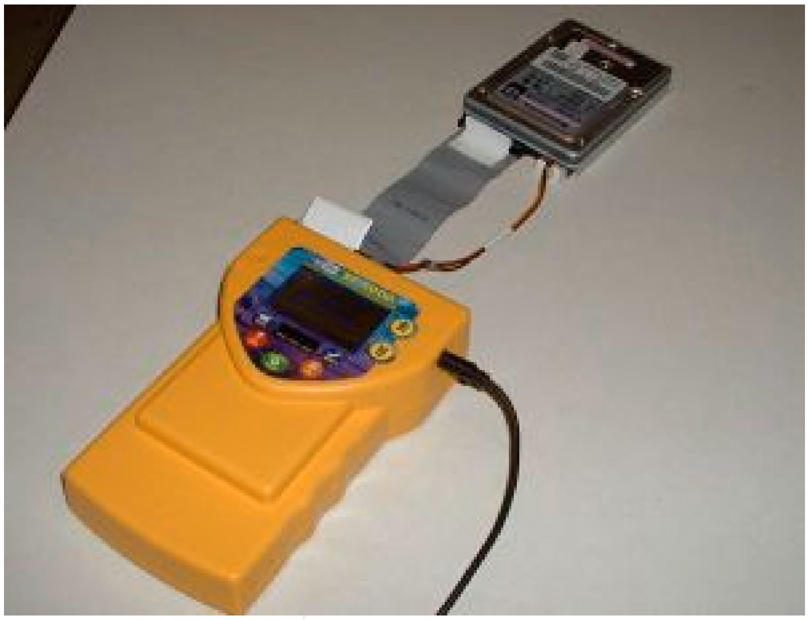


```
[root@localhost root]# fdisk -l

Disk /dev/hdc: 41.1 GB, 41110142976 bytes
16 heads, 63 sectors/track, 79656 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1   *         1       203    102280+  83  Linux
/dev/hdc2           204     77576  38995992   83  Linux
/dev/hdc3         77577     79656   1048320   82  Linux swap

Disk /dev/hdd: 3166 MB, 3166765056 bytes
128 heads, 63 sectors/track, 767 cylinders
Units = cylinders of 8064 * 512 = 4128768 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdd1   *         1        26    104800+   6  FAT16
/dev/hdd2            27       766   2983680   65  Novell Netware 386
[root@localhost root]# md5sum /dev/hdd
f77f6696389c62434b36c3c534aaa15e  /dev/hdd
[root@localhost root]#
```

The resulting MD5 hash is a unique signature of the entire hard drive contents.

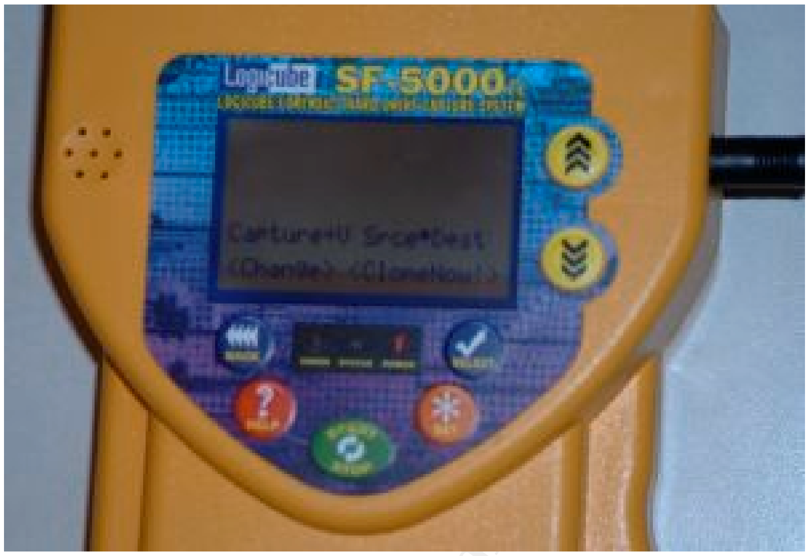
```
Note: The suspect drive was never mounted so it could never
be written to.
```

3. **Capture the suspect drive --** using the Logicube SF-5000u unit
    a. Install the destination drive inside the SF-5000u jumpered as master. Remember, this is the sanitized 80Gb drive from step 1.
    b. Connect the suspect drive externally to the SF-5000u. Using the 40-pin IDE cable and power cable.



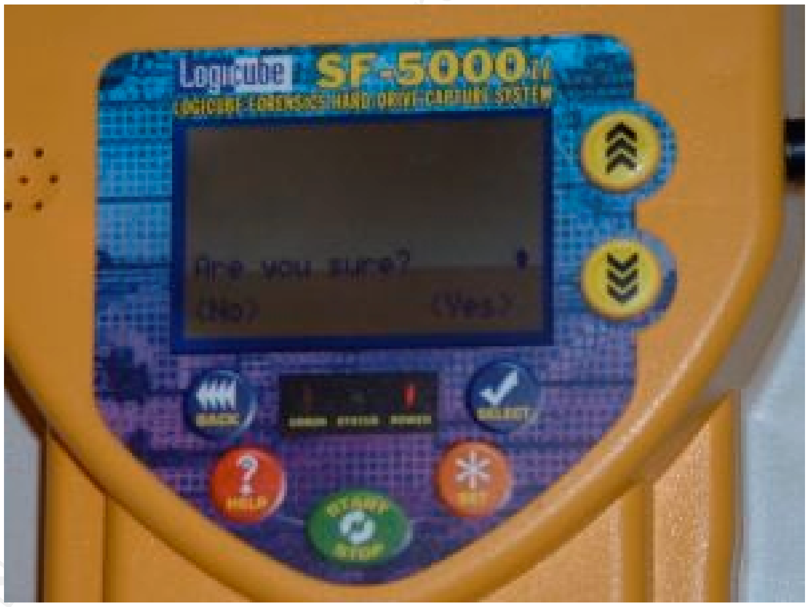
    c. Be sure the SF-5000u recognizes the suspect drive and displays the expected number of sectors.

d. Press the green Start button



Capture+U Srce*Dest
(Change) (CloneNow)

e. Confirm overwrite of destination drive



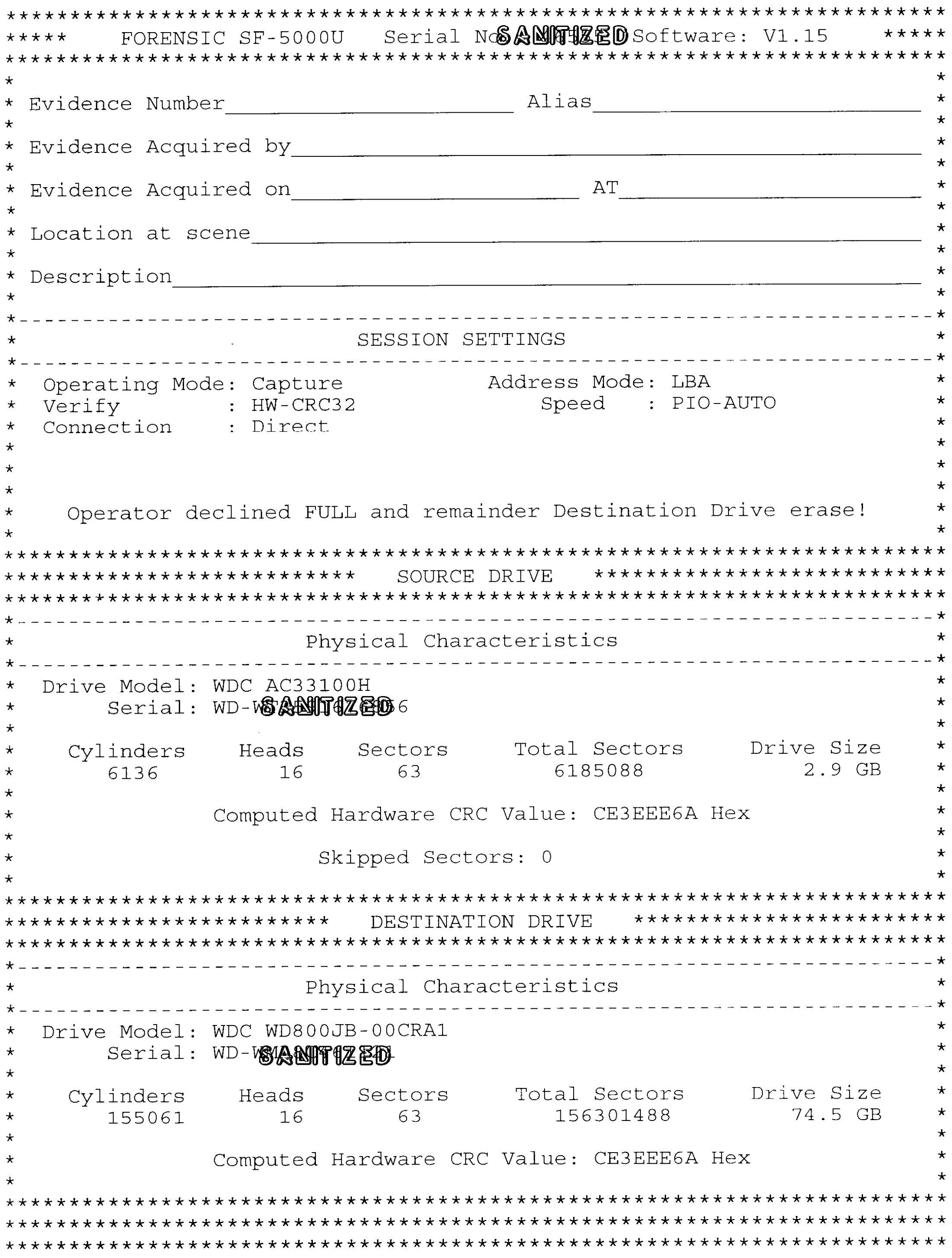
Are you sure?
(No) (Yes)

f. Copying begins.

g. Upon completion the unit displays a confirmation message of an error-free capture.

h. This 3.1Gb drive took approximately 12 minutes to capture.

i. Optional capture session report. The report can be printed on any parallel printer that accepts raw ACSII characters.



```
*********************************************************************
*****    FORENSIC SF-5000U   Serial No SANITIZED Software: V1.15    *****
*********************************************************************
*                                                                   *
* Evidence Number______________________ Alias_________________________ *
*                                                                   *
* Evidence Acquired by_________________________________________________ *
*                                                                   *
* Evidence Acquired on______________________ AT_______________________ *
*                                                                   *
* Location at scene____________________________________________________ *
*                                                                   *
* Description__________________________________________________________ *
*                                                                   *
*-------------------------------------------------------------------*
*                          SESSION SETTINGS                         *
*-------------------------------------------------------------------*
*  Operating Mode: Capture            Address Mode: LBA             *
*  Verify        : HW-CRC32               Speed    : PIO-AUTO       *
*  Connection    : Direct                                           *
*                                                                   *
*                                                                   *
*                                                                   *
*    Operator declined FULL and remainder Destination Drive erase!  *
*                                                                   *
*********************************************************************
**************************    SOURCE DRIVE   ************************
*********************************************************************
*-------------------------------------------------------------------*
*                      Physical Characteristics                     *
*-------------------------------------------------------------------*
*  Drive Model: WDC AC33100H                                        *
*       Serial: WD-WSANITIZED6                                      *
*                                                                   *
*    Cylinders    Heads    Sectors     Total Sectors     Drive Size *
*       6136        16        63          6185088           2.9 GB  *
*                                                                   *
*              Computed Hardware CRC Value: CE3EEE6A Hex            *
*                                                                   *
*                      Skipped Sectors: 0                           *
*                                                                   *
*********************************************************************
************************   DESTINATION DRIVE   **********************
*********************************************************************
*-------------------------------------------------------------------*
*                      Physical Characteristics                     *
*-------------------------------------------------------------------*
*  Drive Model: WDC WD800JB-00CRA1                                  *
*       Serial: WD-WSANITIZED                                       *
*                                                                   *
*    Cylinders    Heads    Sectors     Total Sectors     Drive Size *
*      155061       16        63         156301488          74.5 GB *
*                                                                   *
*              Computed Hardware CRC Value: CE3EEE6A Hex            *
*                                                                   *
*********************************************************************
*********************************************************************
*********************************************************************
```

You'll notice that the Logicube SF-5000u does a CRC-32 error check instead of a MD5 hash. This may bring up some suitability questions such as.

- "CRC-32 is not strong enough."
- "CRC-32 can be broken."
- Or a real geek will say something like "I can alter your data and your CRC-32 value won't even change."

OK, these things may be true, but here's how I look at it:

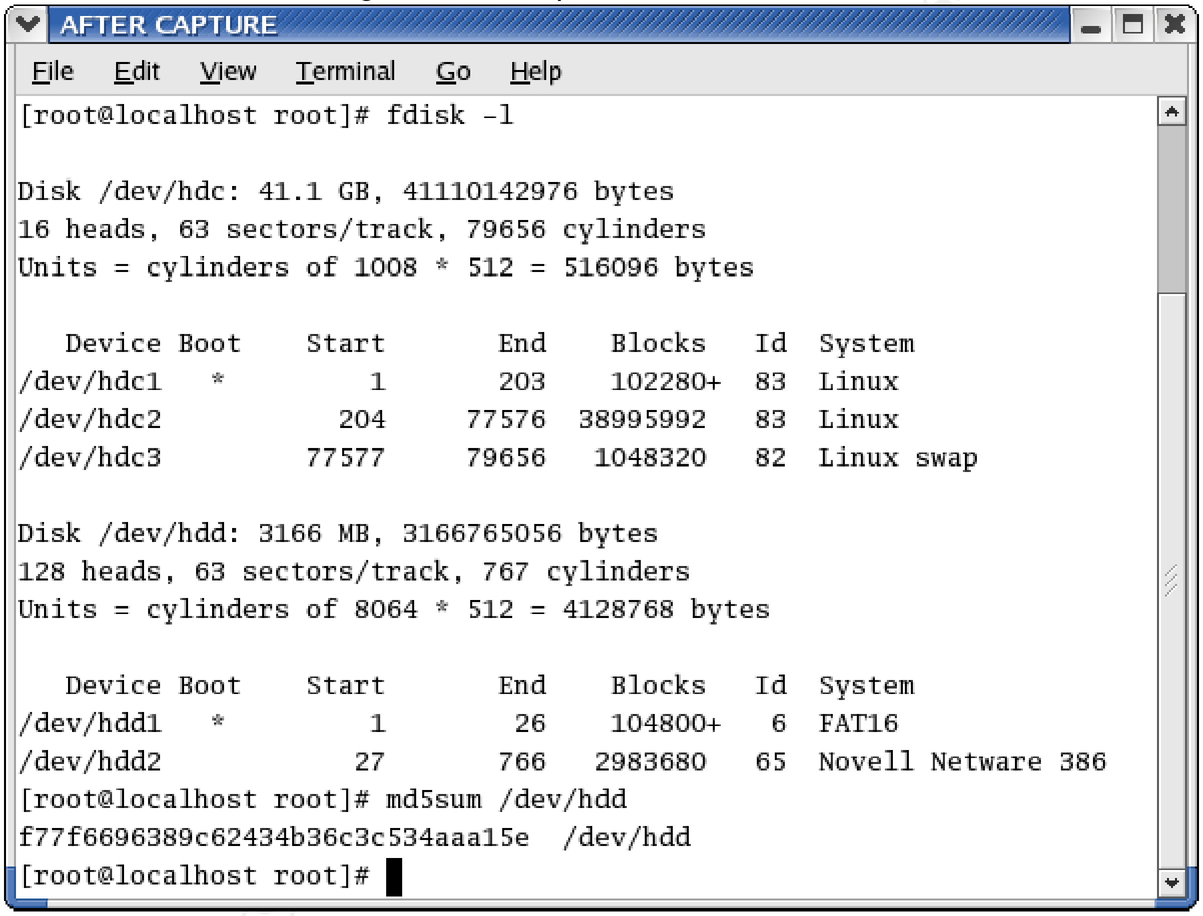
MD5 hashes and CRC-32 are used for different purposes. MD5 is used to authenticate a copy. Such as "You have the same hard drive contents as I do." While CRC-32 is used verify data transmission. As in, during an exchange of data: "here's some data and a checksum to verify".

When we are performing field acquisitions using the SF-5000u, we physically have the original hard drive in our hands -- CRC-32 verifying the transmission of data. So if the source drive is the original and the transmission is verified isn't the resulting copy authentic?

It's not like our skeptical geek can hack the data transmission and alter the data on the fly.

So while I wouldn't want my banking information sent across the public Internet with CRC-32 checksum encryption, I believe it's good enough for validation of a direct-connected data transmission in this scenario.

4. **Get a post-capture signature of suspect drive –** Repeating the same process as in step 2, I will obtain an MD5 hash of the suspect drive.
    a. Using a removable IDE drive bay, I installed the suspect drive in the lab PC jumpered as a slave drive.
    b. Boot into Linux.
    c. Use **`fdisk -l`** to list the partition tables of the hard drives.
    d. Run **`MD5SUM`** against the suspect drive

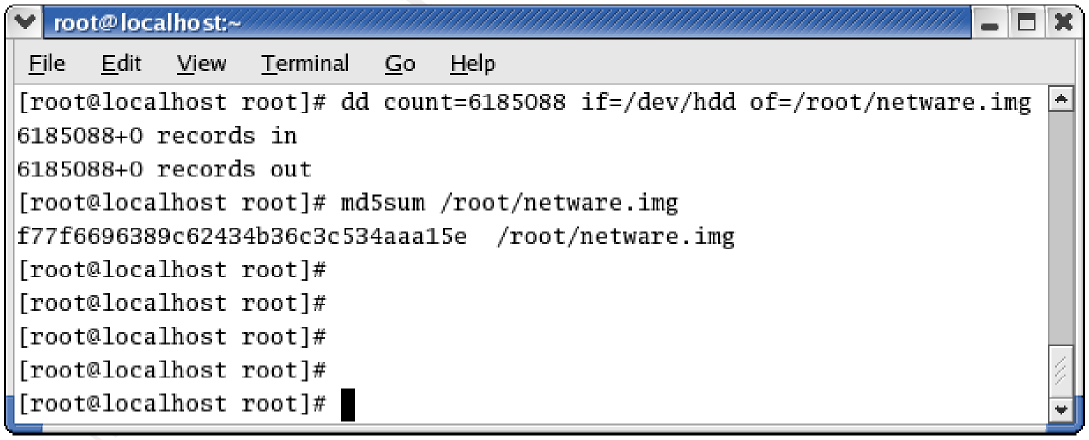


```
[root@localhost root]# fdisk -l

Disk /dev/hdc: 41.1 GB, 41110142976 bytes
16 heads, 63 sectors/track, 79656 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1   *         1       203    102280+  83  Linux
/dev/hdc2           204     77576  38995992   83  Linux
/dev/hdc3         77577     79656   1048320   82  Linux swap

Disk /dev/hdd: 3166 MB, 3166765056 bytes
128 heads, 63 sectors/track, 767 cylinders
Units = cylinders of 8064 * 512 = 4128768 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdd1   *         1        26    104800+   6  FAT16
/dev/hdd2            27       766   2983680   65  Novell Netware 386
[root@localhost root]# md5sum /dev/hdd
f77f6696389c62434b36c3c534aaa15e  /dev/hdd
[root@localhost root]# 
```

```
As a double-check I ran another MD5 hash using EnCase's DOS boot disk:

The hash value of 1 from sector 0 to 6185087 is:
F77F6696389C62434B36C3C534AAA15E
Completed on: 12/12/2003 04:15:47am  Elapsed time: 0:08:18
```

5. Since the pre- and post-capture MD5 hash values from the suspect drive match, we can be assured that the contents of the hard drive has not changed during the SF-5000u capture process.

6. Now we must verify that the SF-5000u made an exact copy of the suspect drive on to the destination drive.
   a. **BUT WAIT! –** The destination drive is an 80Gb drive and the suspect drive is a 3.1Gb. How will the MD5 hashes match?
      i. The SF-5000u places data at the first sector of the destination hard drive and continues sequentially until all the data is copied. So all we have to do is MD5-hash the first 6,185,088 sectors, which is the size of the suspect drive. The remaining sectors on the destination drive will be ignored. We'll make a dd image file of the first 6,185,088 sectors then run an MD5 hash on that file.
      ii. Using a removable IDE drive bay, I installed the destination drive in the lab PC jumpered as a slave drive. Remember first 6,185,088 sectors of the destination drive now contain the data from the suspect drive.
      iii. Boot into Linux.
      iv. Run `dd count=6185088 if=/dev/hdd of=/root/netware.img` this will create the file, netware.img which will consist of the first 6,185,088 sectors of the destination drive.
      v. Run a MD5SUM on the file netware.img file.



```
[root@localhost root]# dd count=6185088 if=/dev/hdd of=/root/netware.img
6185088+0 records in
6185088+0 records out
[root@localhost root]# md5sum /root/netware.img
f77f6696389c62434b36c3c534aaa15e  /root/netware.img
[root@localhost root]#
[root@localhost root]#
[root@localhost root]#
[root@localhost root]#
[root@localhost root]#
```

```
Again, as a double-check I ran MD5 hash using EnCase's DOS boot disk.
EnCase counts sectors starting at 0 thru 6,185,087

The hash value of 1 from sector 0 to 6185087 is:
F77F6696389C62434B36C3C534AAA15E
Completed on: 12/12/2003 04:35:48am  Elapsed time: 0:02:20

Note the faster MD5 hashing elapsed time for this 7,200 RPM ATA/100
drive.
```

7. Since the MD5 hash of the dd image file matches the MD5 hash of the suspect drive we are assured that the SF-5000u made an exact bit-for-bit copy of the suspect drive onto the destination drive.

The fifth criteria for approval is that the drive capture speed be roughly comparable to direct drive-to-drive methods. Step 3(h) I noted that the SF-5000u took about 12 minutes to complete the capture. As a comparison, I placed the original suspect hard drive in my Linux workstation and performed a dd capture of the entire drive with following statement: **`dd if=/dev/hdd of=/root/netware.img`** This process completed in about 17½ minutes. I'm not going to spend a lot of time trying to control the test or over-analyze the results. There are too many variable factors such as disk controller parameters and Linux OS environment configuration such as RAM size, cache and running processes. The only thing I'm trying to do in this section is to get a "reality check" on speed. I want to know that a significant amount of not time is wasted using the SF-5000u in the field versus returning the drive back to the lab for acquisition. As it is, when you get back to the lab you're probably going to re-acquire the destination drive into you forensics PC for analysis.

Analysis -- An investigator can be assured that the SF-5000u can capture a bit-for-bit exact copy of a suspect's hard drive. The built-in disk-wiping feature with signature flag gives the investigator confidence that the destination drive had been previously sanitized and is ready to use. Also, after the capture the SF-5000u offers to wipe any unused space on the destination drive. These features along with the built-in CRC-32 error checking and reporting features separate this unit from other simple cloning devices that are intended for IT staff functions.

If I had unlimited resources, I'd like to test various hard drive interface types such as ATA/66 and ATA/100 along with a Serial ATA converter. Also, I'd like to see how the unit handles hard drive errors. By default, the SF-5000u aborts when it encounters a bad block on the source drive. There is a setting to change that behavior to retry or skip.

Presentation – The validation of this device depends greatly on the authentication of data using MD5 hashing. If I were presenting this validation in court I would need to describe the Message Digest 5 algorithm and its ability to generate unique values for given data. I'd mention the possibility that 2 different data streams resulting in the same MD5 hash is approximately 1 in 2^{128} which is approximately 340 billion, billion, billion, billion.

Putting it in more understandable terms (for a jury):

> Considering the odds of winning the Multi-State Powerball lottery are 1 in 120,526,770, (see http://www.powerball.com/powerball/pb_prizes.asp)
>
> So divde 2^{128} by 120,526,770 and one can conclude that you could win the Powerball lottery 2,823,292,841,257,908,624,477,156,464,342 times before 2 different files would have the same MD5 hash.

Also I would demonstrate that imaging a hard drive using other methods produced the same results.

Conclusion -- I believe the Logicube SF-5000u is a sound forensics data acquisition tool. My test demonstrated that the unit copied the data and did not alter the original.

Its portability and self-contained aspect should be its most favored attributes for field investigators. Logicube offers several optional attachments such as USB, PCMCIA, and Serial ATA hard drive adapters making the unit extremely flexible. Since the device is operating system independent, the user never needs to worry about drivers and compatibility.

For industry standardization I'd like to see an MD5 hash verification and some non-volatile memory to hold the last session's report in memory so those of us that don't travel with a printer can print the report back at the lab.

I have even used the device in the field to capture a 2½" laptop hard drive using an IDE adapter with no trouble.

Since this is a hard drive acquisition device that must be physically connected to the source hard drive, it's not practical to use in the case of a live compromised system that can't be taken down.

Additional Information – The "catch-22" of acceptance is: the more a device gets used and accepted in the industry, the more it will get used and accepted. Also, momentum for a product's acceptance is increased by positive publicity.

The FBI used a Logicube disk-duplicating device in the case of suspected-terrorist Zacarias Moussaoui. In court documents, the government states “Logicube was selected to duplicate the University of Oklahoma hard drives because of its portability. Sewell Affidavit at 3-5, 18. Like Safeback, Logicube has been verified by both its manufacturer and the FBI. Moreover, Logicube performs self-checking functions to ensure that the duplicate drive accurately reflects the contents of the original drive. Finally, although Logicube has not yet been reviewed by the NIST, hand-held disk-duplicators such as Logicube are widely accepted in the information and forensic communities”

PC Magazine®, a computer industry trade magazine gave the SF-5000u a 4 out of 5 rating in a 9/19/03 review by Oliver Kaven.

Part 3 -- Legal Issues of Incident Handling

By the indications of MP3 and DVD information we'll assume that Mr. Price was distributing these types of files to the public.

MP3 is an audio compression algorithm and file format. Music from a compact disc can be compressed and converted to MP3 file resulting in much smaller file with little loss in quality. First of all, the MP3 format is not illegal however distributing copyright protected music in MP3 format certainly is.

Again assuming that the files were copyright protected, Mr. Price has committed a crime under section 506(a)(1) of Title 17 US Code because he has willfully infringed on a copyright "for purposes of commercial advantage or private financial gain". In his message to Mike, he used the term "order" so financial gain can be argured. Section 506(b) allows the court to order the forfeiture and destruction of any copies "and all implements, devices, or equipment used in the manufacture of the infringing copies". So under a worst-case scenario, the court has the power to seize the computer(s) used to make these illegal copies.

```
(this excerpt taken from http://www.copyright.gov/title17/92chap5.html)
```

Title 17 of US Code, Section 506 Criminal Offenses.

```
(a) Criminal Infringement. — Any person who infringes a copyright willfully
either —

   (1) for purposes of commercial advantage or private financial gain, or

   (2) by the reproduction or distribution, including by electronic means,
   during any 180-day period, of 1 or more copies or phonorecords of 1 or
   more copyrighted works, which have a total retail value of more than
   $1,000, shall be punished as provided under section 2319 of title 18,
   United States Code. For purposes of this subsection, evidence of
   reproduction or distribution of a copyrighted work, by itself, shall not
   be sufficient to establish willful infringement.

(b) Forfeiture and Destruction. — When any person is convicted of any
violation of subsection (a), the court in its judgment of conviction shall,
in addition to the penalty therein prescribed, order the forfeiture and
destruction or other disposition of all infringing copies or phonorecords
and all implements, devices, or equipment used in the manufacture of such
infringing copies or phonorecords.
```

The penalty for Mr. Price's crime as defined in Section 2319 of Title 18, Part 1 of US Code depends on the duration of the illegal activity and possibly the number illegal copies distributed.

```
(this excerpt taken from http://www4.law.cornell.edu/uscode/18/2319.html)

Title 18 of US Code Part I, Chapter 113 Section 2319 - Criminal
infringement of a copyright

(a) Whoever violates section 506(a) (relating to criminal offenses) of
title 17 shall be punished as provided in subsections (b) and (c) of this
section and such penalties shall be in addition to any other provisions of
title 17 or any other law.

(b) Any person who commits an offense under section 506(a)(1) of title 17 -

   (1) shall be imprisoned not more than 5 years, or fined in the amount
   set forth in this title, or both, if the offense consists of the
   reproduction or distribution, including by electronic means, during any
   180-day period, of at least 10 copies or phonorecords, of 1 or more
   copyrighted works, which have a total retail value of more than $2,500;

   (2) shall be imprisoned not more than 10 years, or fined in the amount
   set forth in this title, or both, if the offense is a second or
   subsequent offense under paragraph (1); and

   (3) shall be imprisoned not more than 1 year, or fined in the amount set
   forth in this title, or both, in any other case.

(c) Any person who commits an offense under section 506(a)(2) of title 17,
United States Code -

   (1) shall be imprisoned not more than 3 years, or fined in the amount
   set forth in this title, or both, if the offense consists of the
   reproduction or distribution of 10 or more copies or phonorecords of 1
   or more copyrighted works, which have a total retail value of $2,500 or
   more;

   (2) shall be imprisoned not more than 6 years, or fined in the amount
   set forth in this title, or both, if the offense is a second or
   subsequent offense under paragraph (1); and

   (3) shall be imprisoned not more than 1 year, or fined in the amount set
   forth in this title, or both, if the offense consists of the
   reproduction or distribution of 1 or more copies or phonorecords of 1 or
   more copyrighted works, which have a total retail value of more than
   $1,000.
```

It can also be argued that Mr. Price violated a North Carolina statute when he wiped his hard drive. If the repair and restoration of the “damaged” computer exceeds $1,000 he could be charged with class G felony. Time spent by a technician plus software re-licensing could easily total more than $1,000

```
(this excerpt taken from www.ncleg.net/Statutes/GeneralStatutes/HTML/
ByChapter/Chapter_14.html)
```

North Carolina General Statute, Chapter 14, Section 455 - Damaging computers, computer programs, computer systems, computer networks, and resources.

```
(a)It is unlawful to willfully and without authorization alter, damage, or
destroy a computer, computer program, computer system, computer network,
or any part thereof. A violation of this subsection is a Class G felony if
the damage caused by the alteration, damage, or destruction is more than
one thousand dollars ($1,000). Any other violation of this subsection is a
Class 1 misdemeanor.

   (a1)It is unlawful to willfully and without authorization alter, damage,
   or destroy a government computer. A violation of this subsection is a
   Class F felony.

(b) This section applies to alteration, damage, or destruction effectuated
by introducing, directly or indirectly, a computer program (including a
self-replicating or a self-propagating computer program) into a computer,
computer program, computer system, or computer network. (1979, c. 831, s.
1; 1979, 2nd Sess., c. 1316, s. 20; 1981, cc. 63, 179; 1993, c.
539, s. 294; 1994, Ex. Sess., c. 24, s. 14(c); 1993 (Reg. Sess., 1994), c.
764, s. 1; 1995, c. 509, s. 12; 2000-125, s. 5; 2002-157, s. 5.)
```

Because of insufficient evidence, I don’t believe we could pursue action under the Federal Wiretap Act; which addresses the interception of voice or data in real time. Besides, you have to have a high-profile case with a lot of supporting evidence to get the feds involved.

The presence of NetCat, which reads and writes data across a network, brings to mind the Electronic Communications Privacy Act Title 18 US Code, Section 2701, which addresses illegal access to stored data. However this would be a “stretch” for the ECPA, which is intended to deal with “outside” hacking.

Based on Mr. Price’s activities, we must perform a search of all storage locations he had access to so we can eliminate any remaining copyrighted materials. Also, the number of items and copies involved can have an impact criminal and civil prosecution. To reduce our corporate liability we should demonstrate aggressive action to find and destroy illegal items. With assistance from corporate counsel, we need to address any changes to employee handbooks and Acceptable Use Policies and possibly add a "consent to monitoring" policy so we can actively check for employee misconduct in the future.

If our corporate counsel later decides to pursue the matter, we will need to ensure that this evidence is admissible by following sound forensic procedures:

- Never alter the original data
- Verify and authenticate working copies
- Take notes and secure notes and logs
- Maintain a chain of custody for evidence
- Procedures and analysis must verifiable and repeatable

If Mr. Price were distributing child pornography instead of copyrighted music and videos we would have immediately notified our corporate counsel and the local law enforcement. Chapter 10 of US Code Title 18 discusses Sexual Exploitation and other abuse of Children; specifically section 2252A addresses activities in transferring and possessing images.

Working in cooperation with local law enforcement we would actively search all of our systems for any other child pornography and secure or destroy those images as directed by counsel and/or law enforcement. ***Our liability is in doing nothing.***

References

Rivest, Ronald L. “The MD5 Message-Digest Algorithm” Network Working Group, Request for Comments: 1321, April 1992
URL: http://theory.lcs.mit.edu/~rivest/Rivest-MD5.txt

Chuvakin, Ph.D., Anton. “Linux Data Hiding and Recovery”, LinuxSecurity.com 3/10/2002
URL: http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html

Burford, Sean. “Reverse Engineering ELF Binaries on the x86 Platform”, 2002
URL: www.linuxsa.org.au/meetings/reveng-0.2.pdf

Kruse, Warren G., Heiser, Jay G. Heiser. Computer forensics: Incident response essentials.
Boston, MA: Addison-Wesley, 2002

Hsiao, Aron. “Teach yourself Red Hat® Linux® 9 in 24 hours”
Indianapolis IN: Sams, 2003

“Forensic SF-5000u™ User’s Manual” Chatsworth, CA: Logicube Headquarters

Deering, Brian. “Data Validation Using The Md5 Hash”
URL: http://www.forensics-intl.com/art12.html

Kaven, Oliver. “An Evidence Collection Device” PC Magazine September 19, 2003
URL: http://www.pcmag.com/article2/0,4149,1274523,00.asp

“Government’s Opposition to Standby Counsel’s Reply to the Government’s Response to Court’s Order on Computer and E-Mail Evidence”. U.S. v. Zacarias Moussaoui Criminal No. 01-455-A
URL: http://notablecases.vaed.uscourts.gov/1:01-cr-00455/docs/68092/0.pdf

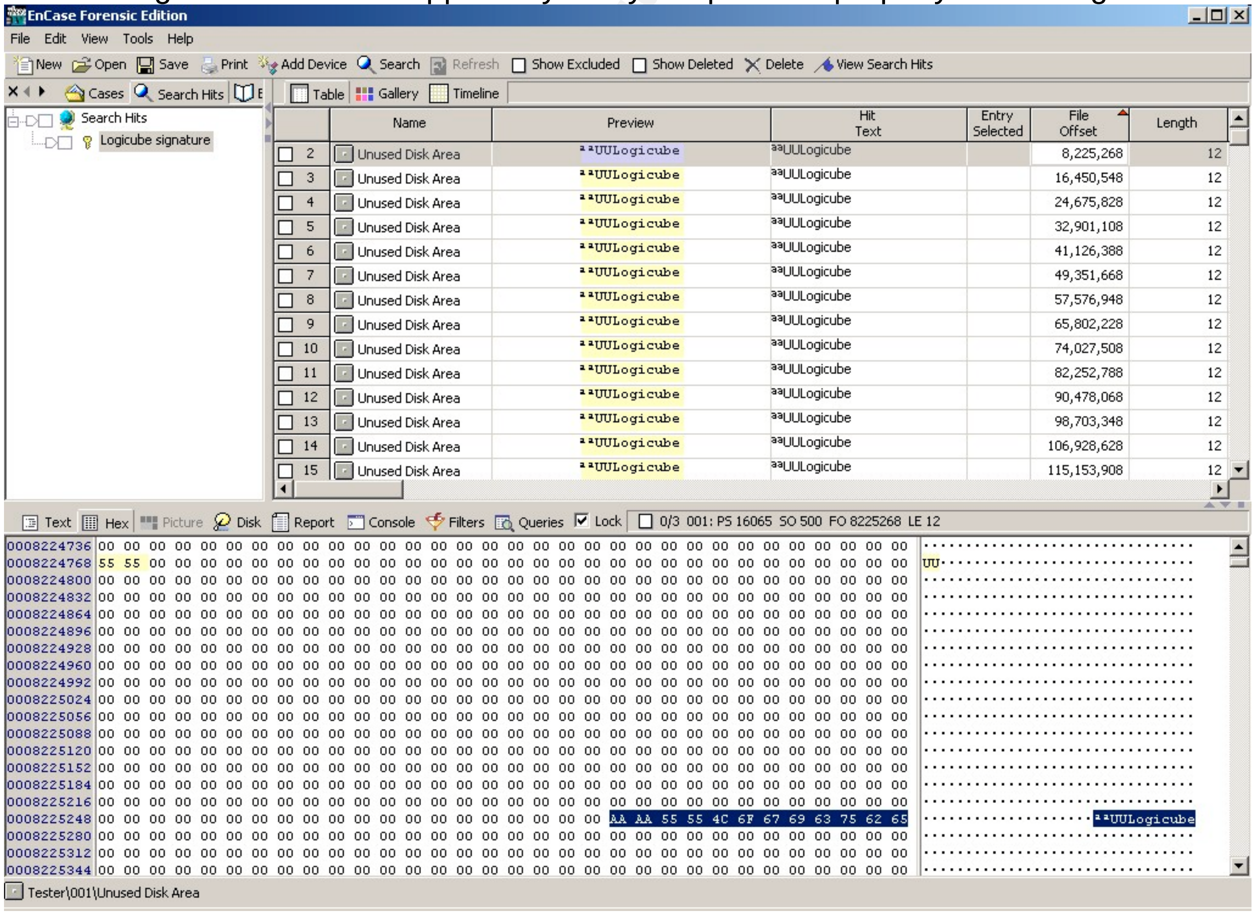
Appendix A -- Logicube SF-5000u WipeClean Signature

The Logicube SF-5000u unit has the ability to wipe a drive clean by writing zeros to every block of the drive. If a source drive is connected to the unit it attempts to find a chunk of empty data (zero's) on the source drive. If at least 16 sectors are found, the unit then streams this empty data onto the destination drive. If no source drive is attached, the unit will write zeros using built-in programmed I/O but this technique is much slower.

Once the drive is wiped, the SF-5000u writes a small signature to the hard drive. The unit later uses this signature at capture time to determine if the installed drive had previously been wiped. If so, wiping again is not necessary and capture can begin.

This small exception to the "wiped-drives-must-be-pristine rule" requires understanding. The SF-5000u user's manual describes the signature as 12 bytes: 0xAA, 0xAA, 0x55, 0x55 followed by "Logicube" at end of the "first sector of each logical cylinder boundary across the entire drive."

Using EnCase software and searching for "Logicube" we see the signatures. Also, we see (2) 0x55 bytes at the beginning of each sector where the signature is written. The 2 additional bytes are not described in the manual but are located on every signature sector. Logicube technical support says they are part of a properly formed signature.


EnCase Forensic Edition
File Edit View Tools Help
New Open Save Print Add Device Search Refresh Show Excluded Show Deleted Delete View Search Hits
Cases Search Hits
Table Gallery Timeline
Search Hits
Logicube signature
Text Hex Picture Disk Report Console Filters Queries Lock

		Name	Preview	Hit Text	Entry Selected	File Offset	Length
☐	2	Unused Disk Area	ªªUULogicube	ªªUULogicube		8,225,268	12
☐	3	Unused Disk Area	ªªUULogicube	ªªUULogicube		16,450,548	12
☐	4	Unused Disk Area	ªªUULogicube	ªªUULogicube		24,675,828	12
☐	5	Unused Disk Area	ªªUULogicube	ªªUULogicube		32,901,108	12
☐	6	Unused Disk Area	ªªUULogicube	ªªUULogicube		41,126,388	12
☐	7	Unused Disk Area	ªªUULogicube	ªªUULogicube		49,351,668	12
☐	8	Unused Disk Area	ªªUULogicube	ªªUULogicube		57,576,948	12
☐	9	Unused Disk Area	ªªUULogicube	ªªUULogicube		65,802,228	12
☐	10	Unused Disk Area	ªªUULogicube	ªªUULogicube		74,027,508	12
☐	11	Unused Disk Area	ªªUULogicube	ªªUULogicube		82,252,788	12
☐	12	Unused Disk Area	ªªUULogicube	ªªUULogicube		90,478,068	12
☐	13	Unused Disk Area	ªªUULogicube	ªªUULogicube		98,703,348	12
☐	14	Unused Disk Area	ªªUULogicube	ªªUULogicube		106,928,628	12
☐	15	Unused Disk Area	ªªUULogicube	ªªUULogicube		115,153,908	12

0/3 001: PS 16065 SO 500 FO 8225268 LE 12

```
0008224736 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008224768 55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 UU······························
0008224800 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008224832 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008224864 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008224896 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008224928 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008224960 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008224992 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008225024 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008225056 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008225088 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008225120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008225152 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008225184 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008225216 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008225248 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AA AA 55 55 4C 6F 67 69 63 75 62 65 ····················ªªUULogicube
0008225280 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008225312 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
0008225344 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ································
```

Tester\001\Unused Disk Area