Trash and Treasure
Computer Forensics and Public Domain Data

By

MICHAEL A. SCOTT

GIAC Certified Forensic Analyst (GCFA)
Practical Assignment Version 1.4
November 2003

ABSTRACT
Computer forensic methods were applied to determine the nature of an unknown binary found on a corporate system. It was proven to be a tool called bmap that was used by an employee to hide data within the hidden slack space of a computer hard drive. The tool showed that a covert text message had been hidden in the system that related to the illegal sharing of media files on the public Internet. In a second case study, a detailed forensic analysis of a computer system left for trash at a public refuse dump was shown to contain critical business and personal data that, despite being deleted by the owners, was easily recoverable. This information could have been used to malicious intent had it fallen into the hands of a hacker. Consequently, methods of securing data and systems are presented in the form of a computer disposal policy. The legal issues of handling the incident in the first case are discussed in relation to UK law.

Conventions

The following conventions are used in this report:
`Monospaced` font indicates computer code, commands/options, hostnames, email/internet addresses and file/directory names. Where commands are shown, **`monospaced bold`** font indicates what the user should type literally. `Monospaced non-bold` indicates the corresponding computer output.

The following prompts indicate**:**

`#`	Shell command is run as the `root` UNIX user
`C:>`	Command is run under Windows

Where Internet URL's are referenced, a direct link to the web page is given within the document body, e.g., [REF].

Footnotes are occasionally used to indicate useful tips or extra information that the reader may find interesting

TABLE OF CONTENTS

1. Introduction

Mention the word forensics to the average 'Joe Public' and it conjures up the images "murders, police and solving crimes". This perception is largely due to media portrayal in shows such as 'CSI: Crime Scene Investigation' where Horatio Caine and his compadres attend dramatic crime scenes, uncovering evidence in order to prove guilt or innocence in exciting crimes. The aim of the forensic investigator is more realistically summarised by accomplished expert witness Jim Bates: "*It is not the job of the forensic investigator to determine innocence or guilt; it is to make the jury understand what happened, and to help the court reach a decision based on the truth"* [JB].

The applications of computer forensics are in fact diverse and may encompass tasks including post-incident root cause analysis, investigating malicious code or investigating serious hi-tech crimes.

The computer forensics investigator has a challenging role requiring rigorous all-round technical IT knowledge as well as 'soft skills' such as the ability to communicate to a non-technical audience and clear and concise report writing on sensitive and complex issues. Similarly to the forensic scientist, the computer forensics expert examines the minutiae from the hi-tech crime scene ensuring that evidence is not invalidated. In essence, the microscopic biological 'DNA' level is replaced by examination of digital bits of information.

A few key general concepts in computer forensics are outlined prior to presenting detailed results.

Computer based evidence

Computer based evidence is that presented in a court of law originating from the original data stored on a computer or storage media. The definition of 'computer' generally includes other hi-tech devices including mobile phones, personal digital assistants (PDA's) and other digital storage devices.

In this study, we describe forensic analysis techniques that comply with the best practice given in the UK Association of Chief Police Officers (ACPO) 'Good Practice Guide For Computer Based Evidence' [ACP].

Of particular importance are the principles that:

- *No action taken by the police or their agents should change data held on a computer or other media which may subsequently be relied upon in Court*

- *An audit trail or other record of all processes applied to computer based evidence should be created and preserved. An independent third party should be able to examine those processes and achieve the same result.*

The last principle refers to maintaining the evidential chain of custody: making sure that the who, what, whereabouts and why of evidence is tracked and recorded throughout its handling.

Hard drive imaging
It is absolutely critical that we don't perform any forensic analysis on the original evidence drive and only on a disk image. To do this, computer software is used to take an identical digital copy of the data contained in the original drive and the drive entered into an evidence bag and kept in a secure place.

Preserving evidence integrity: hash algorithms
Methods must be used by the investigator to ensure that evidence is ultimately admissible in court. To achieve this we create digital 'fingerprints' of the evidence using hashing algorithms. These algorithms have several properties that ensure integrity. Firstly, the same input file will always give the same output and they are practically impossible to reverse [HA]. The hash value output is always the same size regardless of the input size.
The Linux `md5sum` command contains a hashing algorithm that produces a message digest (MD) fingerprint. As a usage example, I created a text file, `forensic.txt`, containing the text "`Sherlock Holmes would have loved computer forensics!`". The hash value was then calculated:

```
# md5sum forensic.txt
# 480b16ae5e7d878c42fe2d13fce9419c      forensic.txt
```

Changing only one letter, '`Holmes`' to '`H0lmes`' produces:

```
# a611aa98181f0f0c7c5a7ef37da826ae      forensic.txt
```

Thus the `md5sum` output is a128 bit (16x8) hexadecimal hash having the important property that even the tiniest changes to the input of the hash function changes the output hash completely.

To get the uniqueness of hashes into perspective consider that a 128 bit signature means that the probability of a hash collision is one in 2^{128} (2 to the power 128): infeasibly small!

Message digest hashes have become the standard method for maintaining integrity of evidence and files downloaded via the Internet. Any file corruption in transit would be clearly reflected in the hash value.

Hard disk fundamentals
Despite the fact that computer hard drives are the 'bread and butter' of computer forensics, their fundamental workings are often neglected in forensic texts. The file structure information we work with is a logical representation of the actual physical drive. A very quick overview of disk basics are presented.

Hard disks read and write data using similar fundamental technology to standard audio and video tapes: ferromagnetic recording.

The advantages of magnetic media storage are :

1. data can be erased and written to quickly and repeatedly
2. the technology is well established and relatively inexpensive
3. magnetic and substrate technology is not very volatile, hence data retention rate is high.

The mechanical 'guts' of hard disks contains highly reflective, rigid platters comprising the magnetic data recording layer deposited on a substrate of precision polished material. The strong thermo-mechanical properties of platters allow high rotation speeds ca. 5000-7200 rpm allowing fast data transfer rates.

Digital information is stored in computer media as a stream of binary '1's and '0''s. Hard drives store this information in the physical form of magnetic domains that can be perceived as rather like minuscule bar magnets with North and South polarity. Electromagnetic drive heads on a rotating arm are positioned above and below the platters to perform the data reading and writing of the magnetic domains. The arm rotates at high rate and the heads move backwards and forwards across the magnetic surface, reading and writing of binary data stored in concentric circular rings called tracks[1].
Within each track exists fixed size sectors: the smallest basic storage unit on a disk that normally occupy 512 bytes. To increase disk manageability and efficiency, the operating system groups these sectors into chunks called blocks or clusters. The cluster size is filesystem and partition size dependent.

So where does this low level disk information fit into computer forensics? Until the first file is written, the clusters are unallocated by the operating system in the File Allocation Table (FAT). These *unallocated clusters* are not of great interest to the forensics investigator until data has been written to disk. Upon file creation, clusters are allocated in the FAT to store the data. When a file is 'deleted', the clusters allocated to the file are released by the operating system so that new files and data can be stored. However, the data associated with the 'deleted' file may remain in the unallocated clusters that are reassigned by the operating system. Unallocated file space potentially contains whole files, file fragments and temporary files which are created by applications.

Since most files will not occupy an exact integral number of clusters/blocks, there is significant *slack space* that exists on the disk. This slack space is not used or seen by the operating system but may contain old data if a new file is created on a part of the disk where an old file was. The FAT16 filesystem used in older Windows systems and DOS has a fixed maximum number clusters, 2^{16} (65536), per disk partition. Since smaller disk sizes were common, this leads to large areas of slack space.

[1] When hard drives refer to cylinders, they mean the stacking of tracks across multiple platters when the drive heads are aligned

Another consideration relevant to forensics is that, although optimal file storage would be in a contiguous data series or fragment, normal file operations produces file allocation to whatever free clusters are available throughout the drive. This process leads to *fragmentation* of clusters. Fragmentation reduces the read/write efficiency of the hard drive since the drive heads must do more work across the drive looking for data in the cluster chain, thereby introducing latency. Forensic tools follow this chain of clusters in order to reconstruct the entire contents.

Open source forensics

All of the forensic tools used in this study were open source and available free of charge for download on the Internet. These include The Coroner's Toolkit (TCT) written in 2000 by Farmer and Venema [TCT]. TCT is a set of UNIX-based command line tools for analysing UNIX filesystems and acquiring data from live systems. Forensic tools were previously the domain of law enforcement agencies only and the development of TCT allowed public access to forensic tools for the first time. However, TCT is limited by working at a low level in the files structure and does not consider human-readable file/directory names. Also, the analysis system has to be of same platform as the suspect system. The development of the tools TCTUTILS and the @stake Sleuth Kit (TASK) overcame these problems [TCT2,SK].

TASK uses a layered model that provides the forensic investigator with access to all elements of the filesystem. The development of Autopsy 'forensic browser' was a leap forward for open source computer forensics as it encapsulated the command line tools in a user-friendly graphical web interface [AT].

2. Method: The Analysis Systems

Preparing the forensic workstations

Two independent forensic workstations were built that have different advantages depending on the type of forensic investigation. The binary analysis workstation was a Toshiba Satellite Pro laptop with a 2.2GHz Intel P4 processor, 60GB Hitachi Travelstar hard drive and 512MB DDR RAM. The fast processor, large disk space and memory combined with the advantage of portability make this ideal for binary analysis.

The forensic media analysis workstation was a self-built tower PC with an Athlon 1800XP processor, 512MB DDR RAM and 120GB Western Digital Hard Disk. This machine has the advantage for post-mortem media analysis that it has four IDE (Integrated Drive Electronics) connectors and the ability to interface to a wide variety of hardware including 3.5" IDE hard drives, 2.5" laptop drives, external Firewire/USB drives and flash memory cards.

In both workstations, the base operating system was constructed from a full installation of Red Hat Linux v9.0. This ensures that all required components and libraries were installed. For forensics Linux is a great choice since it supports a wide variety of partitioning schemes and most importantly allows us to mount a variety of filesystems, regardless of their type and location, onto the local root filesystem. Native Linux support exists for the FAT, Apple Hierarchical (HFS) and UNIX/Linux filesystems. However, most office and home Windows NT/XP/2000 systems encountered use the NT filesystem (NTFS). In Red Hat Linux, there is no native NTFS support and a kernel patch is needed[2].

To obtain the correct patch version, the `uname` is used with the `-srmp` switches to determine the kernel name (`s`), release version (`r`) and machine hardware (`m`) /processor version (`p`).

For the binary analysis laptop with Intel Pentium 4 Processor system we get

```
# uname -srpm
Linux  2.4.20-8 i686 i686
|-s--|----r----|-m-|--p--|
```

We can then download the appropriate NTFS patch for the hardware and kernel versions. In this case, the Red Hat Packet Manager (RPM) file, `kernel-ntfs-2.4.20-8.i686.rpm`, was downloaded from the Linux-NTFS project website [LNT]. The RPM takes care of the installation process and dependencies needed, providing a convenient way of installing pre-compiled binaries. This is particularly useful patching. Usage is straightforward:

```
#  rpm -iv kernel-ntfs-2.4.20-8.i686.rpm
```

where the `-iv` switch installs the kernel patch in verbose mode.

It is worthy of note that the patch allows only *read-only* access to the NTFS mount. In fact, this is more desirable in forensic work as there is no risk of overwriting or corrupting evidence.

Linux contains inherent binary commands that are useful in forensics inlcuding `strings`, `grep` and `file` that will be used frequently. If space is at a premium, a good choice for the base forensic workstations is the Red Hat Workstation installation, primarily intended for software development and contains the libraries and compilers needed for forensics work.

The open source forensics tools described above were installed on top of the base OS and the KHexEdit hexadecimal editor was installed for file analysis [HEX]. A set of network analysis tools such as Snort and Ethereal were installed to perform network forensics [SNO,ETH].

---

2 A computer system at its most basic includes a set of programs called the *operating system* and the most important program in this set is called the *kernel*. It is loaded into memory when the system boots and contains many critical procedures that are needed for the system to operate. The essential capabilities of the system are determined by the kernel.
See 'Understanding the Linux Kernel' for a detailed treatment [KER].

Now that we have the basic forensic toolkits, it is useful to have a clean backup set of forensic tools and Linux binaries that can be run on a CD-ROM and used for remote forensic investigations.

Sterilising the media

Prior to drive imaging on the media analysis workstation, the Linux ext2 partition, `/dev/hda5`, where evidence is to be stored was wiped using the `dd` command to avoid any evidence contamination. Using `dd` makes an exact bit-for-bit copy of an input file (`if`) to an output file (`of`). If the input is set to `/dev/zero`, the zero-filler buffer, and the output to the evidence image partition `/dev/hda5`, then command:

```
# dd if=/dev/zero of=/dev/hda5
```

will write "00 00…" to every block of the partition, rendering it free from any existing data fragments or "sterile".

Safeguarding the analysis systems

Prior to any forensic analysis, it is important to make sure that the analysis systems are themselves pristine to ensure that defending counsel in court cannot accuse the examiner of corrupting the evidence in the course of the analysis.
This was achieved by clean installations the operating systems on new hard drives. The operating system install ISO image files were checked for integrity after downloading from the Red Hat servers. The systems were never connected to the Internet to avoid potential network-based compromises.

As a further 'paranoid' measure, Tripwire® was installed and executed following the guidelines given by Stancin [TW,AS]. Tripwire® is a file integrity tool that creates a baseline of hashes values for the critical system binaries and detects any changes in the event of a system compromise. The program signals that the 'tripwire' has been broken and alerts the user. The `cron` command can be useful to run integrity checks at scheduled intervals that can be emailed.
At a minimum, if Tripwire® is not used, the forensic investigator should keep a backup copy of the md5 hashes of these critical binaries contained in `/usr/bin`, `/usr/sbin`,`/sbin` directories and have clean backup copies to facilitate restoration.

Creating virtual machines using VMWare Workstation

VMWare is a phenomenal tool for computer forensics and software testing. It is a virtualisation tool that allows simultaneous booting and running of multiple operating systems on Intel x86 hardware [VM]. IN the VMWare scheme, the base OS is called the host and additional operating systems are the guests. The guests exist as virtual machines (VM's) within one physical machine and full network functionality is made available. Using 'host-only networking' provides communication only between the guests and host, creating what is, in essence, an isolated network lab.

The VMware Workstation v.4.0 RPM, `VMware-4.i386.rpm`, was downloaded from the VMWare website and licensed.

The installation is trivial:

```
# rpm -ivh VMware-4.i386.rpm
```

Next, the Perl configuration script was executed

```
# vmware-config.pl
```

and following the instructions, the system was configured for '*host-only networking*'. VMware networking behaves like a physical hub: a shared media device that replicates the electrical signals from one device to another. This means that each device on the hub-connected network sees all network traffic. Thus any network traffic on the isolated host-only VMWare network can be captured using network packet 'sniffers'. In unknown binary analysis, this functionality is crucial particularly when malware is involved.

Another useful attribute of VMWare is the ability to easily make copies of VM's useful for restoring backup copies in case of corruption. To make a new Windows 2000 VM, the relevant VM directory containing the `*.vmx` and `*.vmdk` files is copied to a new one:

```
# cp /root/Win-2000 /root/Win-2knew
```

Pristine VM's of Windows 2000 Professional and Red Hat Linux 8.0 were created using the simple on-screen instructions on both workstations. The Windows 2000 VM was installed on an NTFS partition with network workgroup set to "`Analysis`". The network IP addresses were `192.168.0.5` and `192.168.0.7` for the Windows and Linux VM's, respectively. The subnet masks used were `255.255.255.0`.

To get hosts and guest conversing in the isolated network, the Linux `/etc/hosts` and Microsoft Windows 2000 `C:\Windows\System32\drivers\etc` files were edited to map each IP address to a hostname:

```
127.0.0.1       localhost.localdomain   localhost
192.168.0.1     linuxanalysis
192.168.0.5     winanalysis
192.168.0.7     linuxguest
```

where `linuxanalysis`, `winanalysis` and `linuxguest` are the hostnames for the Linux Host, Windows 2000 VM, and Linux 8.0 guest, respectively.

To test the network setup, `ping` is used to check if a given system is alive and responding to the network. Pinging sends out stream of ICMP echo request packets to the remote system which, if alive, reports back with ICMP echo responses. In Linux pressing `<ctrl>-C` will stop pinging.

The `ping`[3] command was run from guests to host and vice-versa as follows:

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\>ping linuxanalysis
Pinging linuxanalysis [192.168.0.1] with 32 bytes of data:

Reply from 192.168.0.1: bytes=32 time<10ms TTL=64
Reply from 192.168.0.1: bytes=32 time<10ms TTL=64
Reply from 192.168.0.1: bytes=32 time<10ms TTL=64
Reply from 192.168.0.1: bytes=32 time<10ms TTL=64

Ping statistics for 192.168.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

------------------------------------------------------------
# ping winanalysis
PING winanalysis (192.168.0.5) 56(84) bytes of data.
64 bytes from winanalysis (192.168.0.5): icmp_seq=1 ttl=128
time=0.664 ms
64 bytes from winanalysis (192.168.0.5): icmp_seq=2 ttl=128
time=0.365 ms
64 bytes from winanalysis (192.168.0.5): icmp_seq=3 ttl=128
time=0.312 ms
--- winanalysis ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2017ms
rtt min/avg/max/mdev = 0.312/0.447/0.664/0.154 ms
```

The same was observed for pinging from the Linux guest. Now we have an isolated network lab that can be useful for containing and observing malware. We need not worry about damaging production systems or other networked systems when investigating unknown files with potentially damaging effects.

A screen capture of the Red Hat Linux 8.0 and Windows 2000 Professional VM guests running on the forensic media analysis workstation is shown in Figure 1.

---

[3] Occasionally `ping` communication between virtual and physical hosts in VMWare goes dead. This appears to be due to the simultaneous presence of NAT (`vmnet8`) and host-only (`vmnet1`) virtual networks. To remedy this, use the `ifconfig` command to take down `vmnet8` as follows:

```
# ifconfig vmnet8 down
```

Alternatively, edit the network configuration in `vmware-config.pl` to remove `vmnet8` completely if NAT networking is not needed.

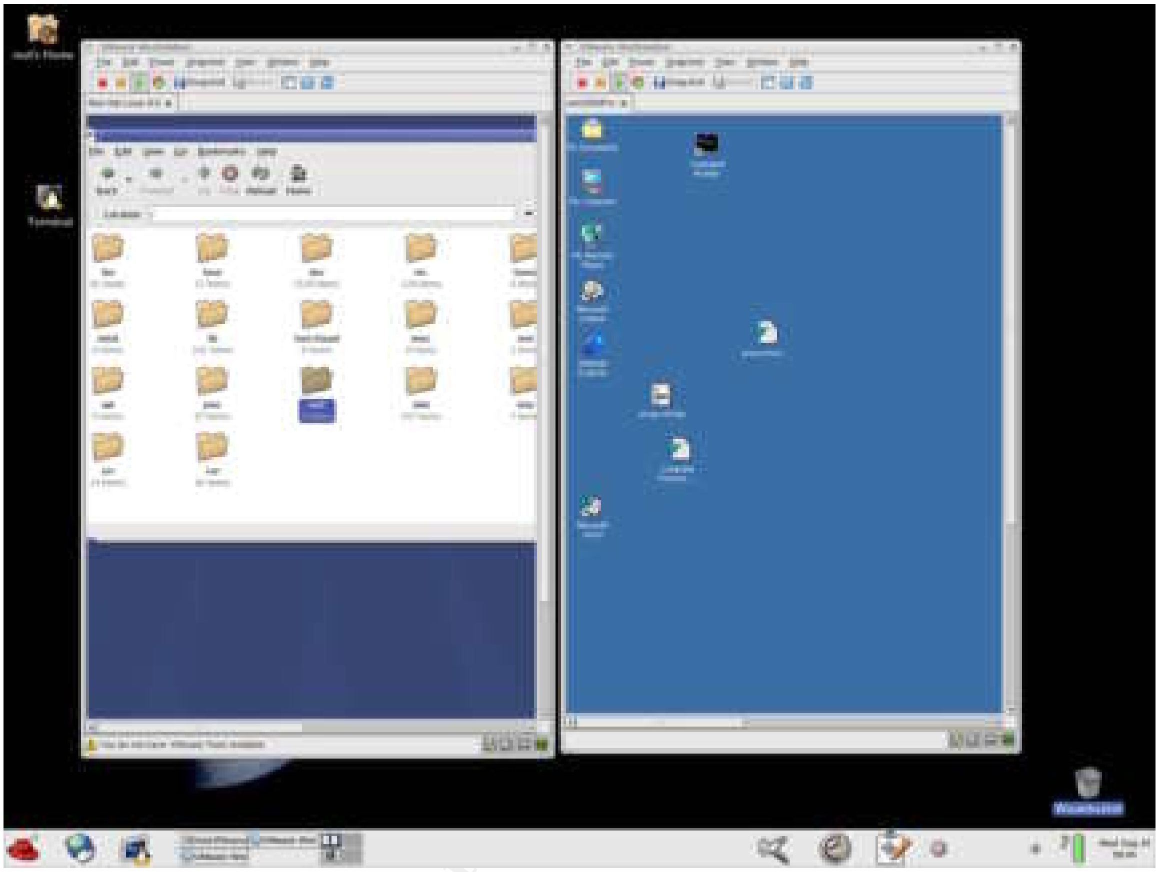


Figure 1: Red Hat 9.0 media analysis workstation host running VMWare Workstation 4, showing Red Hat Linux 8.0 and Windows 2000 Professional guest operating systems running simultaneously with the host

3. Part I: Case #1 - Analysis of an unknown binary

Case #1 background

An employee, John Price was suspended after an internal security audit discovered that he was using the company's computing resources to illegally distribute copyrighted material. Unfortunately, Mr. Price had wiped the hard disk of his office PC before the forensic investigations had begun.
All was not lost though, as a single 3.5" floppy disk was found in the drive of the PC. Although Mr. Price has subsequently denied that the floppy belonged to him, it was seized as evidence and tagged as recorded in Table 1.

Tag ID #	Description
fl-160703-jp1	**3.5 inch TDK floppy disk left in suspect's PC drive unit**

Table 1: Tag ID # and description of floppy disk left in suspect John Price's computer system

The floppy disk was identified at seizure time to contain a number of files, including an unknown binary named prog. The nature and purpose of this binary is completely unknown and its identification is central to the forensic investigation to shed light on Mr. Price's activities. It was suspected that Mr. Price may have had access to other computers in the workplace.

At the time of seizure, a bit image copy of the floppy was taken and the md5 hash was calculated to be `4b680767a2aed974cec5fbcbf84cc97a`.

Case #1 post-mortem analysis

In a post mortem analysis we investigate evidence seized from the crime scene back in the lab in order to gain information about the original usage. This analysis is analogous to a forensic pathologist performing an autopsy on a corpse.

The seized evidence was obtained as the file `binary_v1_4.zip`. This appears to be a compressed `zip` archive. However, vigilance is necessary in forensics since the file extensions can be easily removed and faked. Consequently, the `file`[4] command is used to provide file information by analysing the data content and looking for recognisable binary signatures to detect many different file types:

```
# file binary_v1_4.zip
binary_v1_4.zip: Zip archive data, at least v2.0 to extract
```

shows it is indeed a zip archive. A wealth of information about the archive can be obtained using `zipinfo` in long (`-l`) mode:

[4] `file` checks the 'magic' file `/usr/share/magic` to identify some file types. The magic file contains a huge amount of byte signatures for known file types

```
#  zipinfo -l binary_v1_4.zip
Archive:  binary_v1_4.zip   459502 bytes   3 files
-r--------  2.3 unx   474162 bx   458937 defN 16-Jul-03 06:03 fl-160703-jp1.dd.gz
-rw-r--r--  2.3 unx       54 tx       54 stor 16-Jul-03 07:14 fl-160703-jp1.dd.gz.md5
-rw-r--r--  2.3 unx      39 tx       39 stor 16-Jul-03 07:14 prog.md5
3 files, 474255 bytes uncompressed, 459030 bytes compressed: 3.2%
```

To interpret, this archive 459KB in size, was created by `zip` version 2.3 on (UNIX/Linux) and contains three compressed files. The binary gzipped floppy image file, `fl-160703-jp1.dd.gz`, originally 474Kb was compressed using normal deflation methods to 459Kb in size. The image file was last modified on 16-Jul-03 at 06:03. The `binary_v1_4.zip` file was uncompressed using Linux `unzip` with the `-X` option to restore user and group ownership:

```
# unzip -X binary_v1_4.zip
```

The unzipped files were viewed using the 'list' command `ls` with the all files (`-a`) switch [5]:

```
# ls -la
-r--------   1 root root 474162 Jul 16 06:03 fl-160703-jp1.dd.gz
-rw-r--r--   1 root root     54 Jul 16 07:14 fl-160703-jp1.dd.gz.md5
-rw-r--r--   1 root root     39 Jul 16 07:14 prog.md5
```

The `md5sum` hash was calculated from the command line:

```
[root@linuxanalysis orig]# md5sum fl-160703-jp1.dd.gz
4b680767a2aed974cec5fbcbf84cc97a  fl-160703-jp1.dd.gz
[root@linuxanalysis orig]# tail fl-160703-jp1.dd.gz.md5
4b680767a2aed974cec5fbcbf84cc97a  fl-160703-jp1.dd.gz
[root@linuxanalysis orig]#
```

showing equivalence the original file hash. This proves that the floppy image was not corrupted in transit from the evidence to analysis machine.

Next, the gzipped archive was uncompressed using `gunzip`:

```
#  gunzip  fl-160703-jp1.dd.gz
#  ls -l
-r--------     1 root  root 1474560  Jul 16 06:03
```

[5] The `-a` switch is useful as it shows hidden files/directories that start with a period "." that and would be missed using regular `ls`. Using `ls -A shows` "almost all" files, ignoring the "." and ".." directories.

```
fl-160703-jp1.dd
```

we see that the image of the floppy disk was taking using `dd` and zipped as soon as imaging finished.

UNIX systems have a 'superuser' model for security - an all or nothing approach. The model uses the filesystem as the basic tool to enforce security via access privileges. Administrative tasks are performed by the superuser or `root` while all other 'normal' users have restricted access and should be unable to control critical file and system processes. This conforms to the principle of least privileges that users should be allowed minimum privileges for the tasks they need to perform. Obtaining `root` account access or "rooting" is the holy grail to UNIX hackers and the account should be protected at all costs [MAS].

The floppy image has very restrictive file permissions, `-r--------`, meaning that only the `root` owner has read-only access, equivalent to executing the access mode change command `chmod 400 fl-160703-jp1.dd.gz`.

We use the `file` command again:

```
#  file  fl-160703-jp1.dd
fl-160703-jp1.dd: Linux rev 1.0 ext2 filesystem data
```

Hence the raw image data was originally from an Linux ext2 filesystem. To reconstruct the filesystem, we firstly make a mount directory

```
# mkdir /mnt/binary_analysis
```

to hold the reconstructed file structure. Linux allows us to mount entire disk images onto the local filesystem using the mount command with the loopback option (`-loop`)

The image is mounted with the further constraints to maintain evidence integrity that there is read-only (`ro`) access, no execution of binaries (`noexec`) and the access time is not changed (`noatime`) using the syntax:

```
#  mount -t ext2 -o loop,ro,noexec,noatime     \
fl-160703-jp1.dd   /mnt/binary_analysis/
```

Now looking into the reconstructed directory structure:

```
# ls -la  /mnt/binary_analysis/
-rw-r--r--    1 root   root    2592 Jul 14 15:13 .~5456g.tmp
drwxr-xr-x    2 502    502     1024 Jul 14 15:22 Docs
drwxr-xr-x    2 502    502     1024 Feb  3  2003 John
drwx------    2 root   root    12288 Jul 14 15:08 lost+found
drwxr-xr-x    2 502    502     1024 May  3 11:10 May03
```

```
-rwxr-xr-x     1 502   502    56950 Jul 1415:12  nc-1.10-16.i386.rpm..rpm
-rwxr-xr-x     1 502       502   487476 Jul 14 15:24 prog
```

reveals several interesting files and directories. Amongst these, are the unknown binary prog and nc-1.10-16.i386.rpm…rpm. This looks like the netcat program installation file. Netcat is a client-server tool that allows sharing of files and data across networks [NC].

Note that a hidden temp file `.~5456g.tmp` was uncovered that would have been missed had 'vanilla' `ls` been used.
A more detailed investigation of these files will be performed later in the analysis, after the unknown binary is analysed.

Binary details
Attention is now focused on the prog file. From the `ls` data above, the file size is 487476 bytes and the file access permissions are `-rwxr-xr-x`, i.e., full read/write/execute permissions for the owner with userid (uid) and groupid (gid) 502. This is equivalent to executing `chmod 755 prog`.
In Red Hat Linux, regular, non-privileged user accounts start with uid/gid=500 and increment as more users are added. The system or logical users are assigned uid's 1…499 and 65534, while `root` always has uid=gid=0. The owner of `prog` has uid=gid=502 and the directory named '`John`' has uid 502 ownership. It is a fair assumption that `prog` belong to the suspect John Price who having a regular account with these id numbers. Normally, this information would be available in the `/etc/passwd` file as follows

```
# cat /etc/passwd | grep 502
```

extracting those lines with text including '502'.

It is worth noting at this point that should John Price have had administrator privileges, he would be able to change file/directory ownership.

The `grep` command as used above, is a basic UNIX command that used often, particularly in forensic work. Basically it allows searching for strings within a file, a process referred to as pattern matching.

To verify the integrity of prog again we take the md5 hash and compare it with the original

```
[root@linuxanalysis binary_analysis]#
[root@linuxanalysis binary_analysis]# tail ../../GCFA/analysis/orig/prog.md5
7b80d9aff486c6aa6aa3efa63cc56880  prog
[root@linuxanalysis binary_analysis]# md5sum prog
7b80d9aff486c6aa6aa3efa63cc56880  prog
[root@linuxanalysis binary_analysis]#
```

Clearly, integrity of the file has been preserved from seizure through to analysis.

Each file on a UNIX filesystem has a unique inode value. Inodes are file 'metadata': files that point to a variety of file attributes such as file owner, size and file modification, access and change (MAC) times. These MAC times are very important in computer forensics since they are extremely sensitive to filesystem changes. Even running a single command could change the access or `atime` of a file easily. Hence it is important that the evidence is mounted read-only and MAC times are grabbed as early as possible.
To identify the MAC times of prog we firstly use `ls` again with the `-i` switch to show the associated inode number:

```
#    ls -i prog
     18 prog
```

hence prog is associated with inode 18. Using `debugfs`, the ext2 filesystem debugger with the `stat` option enables examination of inode 18 in the original image. Usage:

```
# debugfs -R "stat <18>"  fl-160703-jp1.dd
debugfs 1.19, 13-Jul-2000 for EXT2 FS 0.5b, 95/08/09
Inode: 18   Type: regular    Mode:  0755   Flags: 0x0
Generation: 414131
User:   502   Group:   502   Size: 487476
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 960
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x3f14eb2d -- Wed Jul 16 07:05:33 2003
atime: 0x3f14ecdd -- Wed Jul 16 07:12:45 2003
mtime: 0x3f12bd00 -- Mon Jul 14 15:24:00 2003
…
```

Thus, prog was last modified (`mtime`) on Mon July 14 2003 at 15:24, last read (`atime`) on Wed July 16 at 07:12 and the inode contents last changed (`ctime`) on Wed July 16 2003 at 07:05.

One caveat is that using the `touch` and `debugfs` commands, it is possible to alter MAC times easily.

Deleted inodes can also be uncovered using **`debugfs`** using the **`lsdel`** option

```
# debugfs -R "lsdel" fl-160703-jp1.dd
debugfs 1.19, 13-Jul-2000 for EXT2 FS 0.5b, 95/08/09
0 deleted inodes found.
```

There are no deleted inodes in the floppy image.

The `strings` command is another great inherent UNIX tool that enables extraction of human-readable text strings within any files including binaries[6].
All text strings from prog were extracting to text file `prog.strings.txt` using the strings command:

```
#  strings prog > prog.strings.txt
```

This file was further examined for keywords that may aid in the identification of the prog binary.
Some useful keywords extracted from prog are:

```
operate on ...
target
write output to ...
outfile
test for fragmentation (returns 0 if file is fragmented)
checkfrag
display fragmentation information for the file
frag
wipe the file from the raw device
print number of bytes available
test (returns 0 if exist)
wipe
place data
display data
extract a copy from the raw device
list sector numbers
operation to perform on files
.
.
version
autogenerate document ...
1.0.20 (07/15/03)
newt
.
.
s fragmented between %d and %d
%d %s
getting from block %d
file size was: %ld
stuffing block %d
%s has slack
```

---

6 For the initial snoop into a large image that will generates a lot of `strings` output, piping (|) the output of strings into a pager like `less` is a good idea. Page-by-page navigation is facilitated and potential crucial information is less likely to be missed. Usage: **`# strings filename | less`**
Also to show strings of character size `n` or above only use: **`# strings –n filename`**

```
%s does not have slack
%s has fragmentation
%s does not have fragmentation
bmap_get_slack_block
```

Program description

Using `file` on prog:

```
#  file prog
prog: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
for GNU/Linux 2.2.5, statically linked, stripped
```

We see that prog is a 32-bit Linux executable, more specifically an Executable and Linked Format (ELF) format file that has been stripped of debugging symbols. The `readelf` command provides more specific information about ELF binaries:

```
#  readelf -a prog
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                             2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x80480e0
  Start of program headers:          52 (bytes into file)
  Start of section headers:          486796 (bytes into file)…
```

These commands show that prog was compiled on Intel 80386 (little Endian) architecture and is statically-linked. Statically-linked files contain all the libraries that they need to run[7].

The entry point address of 0x80480e0 is typically in the range for normal executables, whilst malicious software (malware) can have abnormal entry points.

From the MAC time information given by `debugfs`, it appears that the last time prog was used is given by the last access time (`atime`) of Wed July 16 at 07:12.

---

[7] Had the binary been dynamically linked, the `ldd` or `readelf -d` commands are useful to list the dynamic dependencies of the file showing which system libraries are needed to run the file

Forensic investigation: live analysis

Our static, post mortem analysis has yielded a great deal of information about the seized image file and the prog binary. The only real way to see what the binary does is to run it 'live' in a protected network and perform a behavioural analysis. The binary is treated at this stage as though it was malware that could do anything to our systems. Therefore, prog was executed inside the Red Hat Linux 8.0 VM using host only networking.

To capture potential network traffic, Snort was used [SNO]. Snort is a free, customisable and comprehensive network intrusion detection system (NIDS) that runs under various UNIX flavours, Windows and MacOS X. There are three operational modes: sniffer, packet logger and intrusion detection. In this study we use only sniffer mode to read every packet off the virtual network and dump them into a human-readable form on stdout[8,9]. Usage:

```
# snort -i vmnet1 -vd
```

where `-i`,`-v`,`-d` specifies our virtual network interface, verbose mode and dump packet payloads, respectively.

Packet sniffing was tested by pinging from the Linux VM (192.168.0.7) to the host OS (192.168.0.1) and using the above command. Here is a sample network traffic decoded by snort:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
08/01-20:16:37.912428 192.168.0.1 -> 192.168.0.7
ICMP TTL:64 TOS:0x0 ID:5238 IpLen:20 DgmLen:84
Type:0  Code:0  ID:36355  Seq:16384  ECHO REPLY
2A C3 76 3F B8 12 0E 00 08 09 0A 0B 0C 0D 0E 0F  *.v?...........
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

This shows the header and payload of an ICMP ECHO REPLY packet associated `ping`, verifying that the sniffer is up and running.

Now we execute the `prog` binary inside the Linux8.0 VM:

```
# ./prog
```

8 Sniffing network traffic with the network interface in promiscuous mode requires the packet capture libraries, `libpcap` (for UNIX) and `winpcap` (for Windows) to be installed [LIB].

9 To determine if libpcap is installed in Linux type **`# whereis libpcap`** which should return something like `/usr/lib/libpcap.so.` Also, using: **`# snort -vd | tee /tmp/snort.output`** will output to the screen (stdout) and the file `/tmp/snort.output` at the same time

```
no filename.  Try '--help' for help.
```

we see a banner indicating the usage and help parameters. No output from the snort sniffer is observed meaning that prog has no associated network traffic.
Clearly a target file is needed to operate on. Following the advice and adding the help switches:

```
# ./prog --help
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…
```

The prog binary looks as though it performs and reads files/disk operations.

To test this theory, the text file `gcfatest.txt` was created using

```
# touch gcfatest.txt
```

and executing prog on it with various switches from the help file above will give a more information about prog does

```
# ./prog -m gcfatest.txt
26029336
26029337
26029338
26029339
```

```
26029340
26029341
26029342
26029343
```

According to the help file these are the sector numbers occupied by the `gcfatest.txt` file.

To display 'data about the file' using prog with the `-s` switch gives:

```
# ./prog -s /GCFA/gcfatest.txt
getting from block 3253667
file size was: 43
slack size: 4053
block size: 4096
```

Interesting. Following the earlier discussion of hard disks we see that prog allows us to view the slack space available. In this case the file occupies only 43 bytes and a further 4053 bytes of slack space exists out of the partition block size of 4kb.

The author felt that this behavioural information in combination with the file attributes uncovered in the post-mortem analysis were adequate to successfully identify the true name of prog using the Google web search engine. Besides the abundance of elaborate hi-tech instruments at our disposal, Google remains one of the forensic analyst's most valuable tools!

Using the keywords list, search queries were performed on targets: "bogowipe"; "wipe" and "bmap + slack". The first returned no search results, the second returned the page for 'wipe- A UNIX tool for secure deletion' which was investigated further. The file `wipe-0.19-20030913.tar.gz` was downloaded an investigated. Although it is a useful file deletion tool in itself, wipe, was ruled out as a candidate for prog. The last query found a gem of an article in Linux Security called "Linux Data Hiding and Recovery" by A. Chuvakin [AC]. The article describes methods for data hiding and includes the text " *the obscure tool bmap exists to jam data in slack space, take it out and also wipe the slack space, if needed.*" This sounds like it could be the culprit.

To test this theory, the instructions outlined in the article to use bmap to hide text data within the slack space of a file using the `-p` switch and recover it with `-s` switch were applied directly to the prog binary. I attempted to put the following text into the `gcfatest.txt`:

```
# echo 'Sherlock Holmes would have loved data hiding!' | ./prog
-p /GCFA/gcfatest.txt
stuffing block 3253667
file size was: 43
slack size: 4053
```

```
block size: 4096
```

The large area of slack space (4k) within the block size used by the filesystem is perfectly adequate for hiding a small text message.

To read the hidden message back use:

```
./prog -s /GCFA/gcfatest.txt
getting from block 3253667
file size was: 43
slack size: 4053
block size: 4096
Sherlock Holmes would have loved data hiding!
```

For completeness we read the same slack space data using bmap1.0.20. Using `-s` produces a syntax error and according to the help file the correct syntax is `-slack` for bmap. Thus:

```
#   bmap -slack /GCFA/gcfatest.txt
getting from block 3253667
file size was: 43
slack size: 4053
block size: 4096
Sherlock Holmes would have loved data hiding!
```

Shows unequivocally that prog and bmap are the same program.

The frightening aspect of bmap is that inserting covert messages in slack space subverts our integrity standard, `md5sum`. Hashes were calculated before and after data hiding on `gcfatest.txt`, with no observed change in the hash value `7db58fbda06fa15435d00a569371717d`.

This is because the hashing function operates only on files seen by the filesystem and simply cannot see the data hidden in the slack space. Clearly, this gives a cunning way of hiding small files within the slack space of a filesystem that would not be easily detected by the regular user.
This is an alternative to steganographic methods that often use the least significant bit (LSB) information on image and media files to hide hidden messages. Our visual and aural perception are not sufficiently sensitive to notice the small changes in the LSB information of images and sound files caused by data hiding [ST].

Using this forensic tool, a hacker could hide elements of a rootkit or other secret information (usernames and passwords) inside the slack space using bmap. Additional data encryption could be used to garble a small file or message prior to hiding inside the slack space. In this way, even if a forensic examiner found data in the slack space, it would remain confidential unless the suspect's private/public key was arrested.

Program identification

From the Chukavin article, there is a direct link to the Scyld corporation webpage that contains the bmap source code in both RPM and `tar.gz` formats [SCY]
All available versions of the bmap source code from v1.0.16-v1.0.20 were downloaded.

The prog keywords showed "`prog:1.0.20`" so it seems logical to perform a comparison with bmap v1.0.20. The tar archive was unzipped using

```
#  tar –xzvf bmap-1.0.20.tar.gz
```

and a quick `vi` look at the `bmap.c` file gives:

```
 * bmap.c userlevel blockmap utility for Linux.
 *
 * Maintained 2000 by Daniel Ridge in support of:
 *   Scyld Computing Corporation.
 *
 * The maintainer may be reached as newt@scyld.com or C/O
 *   Scyld Computing Corporation
 *   10227 Wincopin Circle, Suite 212
 *
```

Revealing that bmap is a Linux block mapping utility. The obscure phrase '`newt`' was found in the keyword list and is actually the email handle of the program author Daniel Ridge.

In the standard build, bmap uses a `Makefile` to build the binaries needed. The `Makefile` facilitates installation by specifying dependencies, which files to build and many other build parameters. A quick look through the `Makefile` shows that the program will output the author's name, the program version (1), the patch number (20) and the build date based on the current system date. Referring back to page 21 we see the date of the prog build shown to be is 15th July 2003. This conflicts with the last modified date of 14th July. I suspect that the build date in the `Makefile` was hardcoded to 15th July.

The bmap binary was built in the standard way explained in the `README` file simply by executing:

```
#  make
```

Using :

```
# file bmap
```

shows that the standard build produces a dynamically-linked binary, i.e., one that depends on shared system libraries to run. To investigate these library dependencies we use list dynamic dependencies command (`ldd`) as follows:

```
# ldd ./bmap
libc.so.6 => /lib/tls/libc.so.6 (0x42000000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

ergo bmap depends on two libraries: `libc.so.6`, the Standard C library and `ld-linux.so.2`, a dynamic-linker library.

In forensics and incident handling, 'paranoid mode' is a good stance to adopt and it is a very good idea not to trust unknown binaries, libraries or pre-compiled binaries. This is particularly important when analysing a system that may have been compromised. Hacker rootkits often install trojan system binaries and libraries so that the detection is made more difficult.

To avoid these problems and remove dependency on system libraries, one can create a bespoke set of static binaries using trusted source code from the GNU public library at ftp.gnu.org[10]. The set of binaries can be built and burned onto a CD to make a mobile 'jump kit' that can be taken on investigations. Running jumpkit binaries allows examination of a subverted system and detailed directions on building a jump kit are given in the 'JumpKit_HOWTO' [KIT].

We know that the prog is a static binary, independent of system libraries. To make a static version of bmap we first clean the dynamic build using

```
#  make clean
```

and then edit the `Makefile` to include the '`-static`' flag in the line '`LDFLAGS= -static`'. Rebuilding the binary using and checking the `file` attributes:

```
# make ; file bmap
```

produces the output:

```
bmap: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
for GNU/Linux 2.2.5, statically linked, not stripped
```

It was observed that prog was also stripped of compile symbols to reduce the file size further and help prevent detection signatures. This can be most easily be performed in the `Makefile` by appending the `-s` switch, i.e., '`LDFLAGS= -static -s`'. Alternatively, a post-build `strip` command will achieve the same. We clean the previous build of bmap and recompile and strip using the first method. Using `file` again:

```
# file bmap
```

---

10 The Free Software Foundation co-ordinates the GNU project, the aim of which is to maintain free operating systems and software, freely distributed and usable by the public.

```
bmap: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
for GNU/Linux 2.2.5, statically linked, stripped
```

We have made a stripped, statically-linked version of bmap.Table 2 shows a comparison of the various binaries.

Binary	Linkage	stripped/unstripped	File size / kb
`prog`	static	stripped	487
`bmap1.0.20`	static	unstripped	652
`bmap1.0.20`	static	stripped	544
`bmap1.0.20`	dynamic	unstripped	222

Table 2: Comparison of binary file sizes when compiled with static vs. dynamic and whether compile symbols are stripped or not

It can be seen that prog is some 57kb smaller than smallest static build of `bmap1.0.20` binary is very much smaller than the static binaries.

How do we explain the smaller size of prog? Firstly, prog seems to use different command syntax from bmap v.1.0.20. In prog we use the abbreviated switches e.g., `-s` to show slack space info whilst in bmap we have to specify `-slack`. This could be due to edited source files by the suspect or the use of an earlier version of `bmap` than that found on the Internet of v1.0.16. If prog is not based on bmap v1.0.20 then the version number given in the `Makefile` could have been changed.

It is clear that the source code and/or build parameters have been changed in the creation of prog from the original bmap. Given the fact that md5 hashes are sensitive to the change in just one character, it will be extremely difficult to obtain an `md5sum` and file size match between prog and bmap.

The two calculated hashes are:

```
05a2e13dbd863be8c24e72e074bf347b  bmap
7b80d9aff486c6aa6aa3efa63cc56880  prog
```

show a clear mismatch in the calculated sums and also that the analysis did not change the prog hash value. Despite the fact that the hashes do not match and an exact character-by-character correspondence between prog and bmap was not obtained, the binary analysis performed proves without doubt that they are indeed the same program, performing exactly the same actions.

Forensic details

In general, the installation and execution of any executable, however covert it's operation, will leave evidence footprints the system. These footprints are investigated to ascertain if prog was executed on the system and to aid the Company's systems administrators pursue any further investigations.

The installation and execution procedure would likely have involved the following steps leaving forensic footprints:

(1) **Building of a statically-linked bmap binary**
As we have seen, the standard bmap build generates a dynamically-linked, unstripped executable. The necessary creation of the stripped, statically-linked binary involves editing the `Makefile` to include the `-static` and `-s` flags and would reflect in a change in the `Makefile` last modified date. Also the build process would affect the last access times of the C (`.c`) and header (`.h`) files if the original source code can be located. The bmap executable would also be created on the system. If `make install` was used then, according to the standard `Makefile`, bmap would be installed into the system directory `/usr/local/bin`.

(2) **Copying of bmap to floppy and renaming**
The floppy drive was partitioned with a Linux ext2 filesystem. Therefore, at some point the command `mk2efs` would have been used to create the filesystem. Also the copying of the binary would involve mounting the floppy onto the local filesystem of the suspect's machine in e.g., `/mnt/floppy`. We know that the file was renamed to prog prior to copying.

(3) **Testing bmap/prog**
Usually a newly acquired tool would be tested examine how it works prior to being used in anger. Knowing the nature of `bmap` as a data hiding tool, it is likely that files would exist with data hidden in the slack space similarly to the simple examples given in this paper.

(4) **Program execution**
From the post-mortem forensic analysis, we know that `prog` is a statically-linked ELF binary that contains the libraries it needs to run. As such, it does not depend on any system libraries and leaves a less obvious evidence trail on execution. We have seen using snort that there is no network traffic is associated with the executable.

On execution, bmap/prog looks for a target filename. If there is no target then the program displays a help banner to stdout and then exits. When a target filename is supplied, there is direct effect on the filesystem and the file can have slack space information displayed, stuffed with data or wiped.

The `strace` command is a useful tool that shows system calls made by an executable file. A *system call* is implemented in the Linux operating system kernel. When a program makes a system call, the arguments are packaged up and handed to the kernel, which takes over execution of the program until the call completes. The GNU C library wraps Linux system calls with functions so that they can be called up easily. The input/output functions `open` and `read` are examples of system calls.

Running `strace` as follows:

 # strace –f –o progstrace ./prog –s ../gcfatest.txt

tells `strace` to record the system calls `prog` makes while reading the slack space of `gcfatest.txt`, including all forked child processes (`-f`) that may be started and outputs (`-o`) to the text file `progstrace`. The system calls output for `prog and bmap` produce identical outputs showing again that they are identical.

```
4577 execve("./prog", ["./prog", "-s", "../gcfatest.txt"], [/* 38 vars */]) =
0
4577  fcntl64(0, F_GETFD)               = 0
4577  fcntl64(1, F_GETFD)               = 0
4577  fcntl64(2, F_GETFD)               = 0
4577  uname({sys="Linux", node="linuxanalysis", ...}) = 0
4577  geteuid32()                       = 0
4577  getuid32()                        = 0
4577  getegid32()                       = 0
4577  getgid32()                        = 0
4577  brk(0)                            = 0x80bedec
4577  brk(0x80bee0c)                    = 0x80bee0c
4577  brk(0x80bf000)                    = 0x80bf000
4577  brk(0x80c0000)                    = 0x80c0000
4577 lstat64("../gcfatest.txt", {st_mode=S_IFREG|0644, st_size=43, ...}) = 0
4577  open("../gcfatest.txt", O_RDONLY|O_LARGEFILE) = 3
4577  ioctl(3, FIGETBSZ, 0xbfffd964)    = 0
4577  lstat64("../gcfatest.txt", {st_mode=S_IFREG|0644, st_size=43, ...}) = 0
lstat64("/dev/hda8", {st_mode=S_IFBLK|0660, st_rdev=makedev(3, 8), ...}) = 0
4577  open("/dev/hda8", O_RDONLY|O_LARGEFILE) = 4
4577  ioctl(3, FIGETBSZ, 0xbfffd8d4)    = 0
4577  brk(0x80c2000)                    = 0x80c2000
4577  ioctl(3, FIBMAP, 0xbfffd964)      = 0
4577  write(2, "getting from block 3253667\n", 27) = 27
4577  write(2, "file size was: 43\n", 18) = 18
4577  write(2, "slack size: 4053\n", 17) = 17
4577  write(2, "block size: 4096\n", 17) = 17
4577  _llseek(4, 13327020075, [13327020075], SEEK_SET) = 0
4577  read(4, "Sherlock Holmes would have loved"..., 4053) = 4053
4577  write(1, "Sherlock Holmes would have loved"..., 4053) = 4053
4577  close(3)                          = 0
4577  close(4)                          = 0
4577  _exit(0)                          = ?
```

The trace is relatively easy to interpret with the aid of a system call reference for i386-Linux [SYS]. The salient points are:

(1) The file prog is executed (`execve`) with process id 4577. No other child processes are started using `fork`. The `gcfatest.txt` file is opened
(2) The file descriptor flags are `fcntl64(2, F_GETFD)` gets the file descriptor flags for the open file
(3) Get real and effective user and group id's
(4) Change space allocation for calling process data segment using `brk()`

(5) The file is opened in read-only mode and low level disk functions are called that get the slack and block sizes. The values are then written to stdout (screen) followed by any hidden slack space data. Lastly, the program exits

Again we see no network activity/ processes associated with prog that would be observed clearly in the `strace` output.

Investigation of other evidence found on floppy

The floppy image seized contains several potentially crucial evidence files and directories that need to be analysed further. These may give further leads to the usage of prog and the behaviour of the suspect. _The `strings` command is used again to dig for clues on the image and use the Autopsy forensic browser to analyse the image.

Strings analysis

The `strings` command was executed on the floppy image:

```
# strings fl-160703-jp1.dd
…
vmware
cd ..
vmware-config.pl
vmware
LOGNAME=root
xmms-mpg123-1.2.7-13.i386.rpm..rpmUU
UU a
vmware
cd ..
vmware-config.pl
vmware
LOGNAME=root
…
```

This strongly connotes that the suspect's desktop system may have be running VMWare. This may be used to avoid detection of illegal activities. Xmms is the Linux media player program most commonly used to play digitised audio files.

Autopsy analysis

Both TCT and TCTUtils work from the command line and make it tedious when analysing large image files. Autopsy encapsulates the TCT and TASK commands within a graphical HTML browser interface. The analysis system used Mozilla v1.1 as a web browser which was started automatically when Autopsy was executed. This excellent tool makes it easy to browse images quickly. The image is mounted read-only and Autopsy reads the on-disk structures directly. Therefore, the entire filesystem data is made available to the investigator.

The `fl-160703-jp1.dd` image was mounted in Autopsy by specifying the ext2 partition type and the author initials "MAS" as the investigator. The time zone was set to "CST".
Clicking on the image information button shows full details of the image:

```
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: EXT2FS
Volume Name:
Last Mount: Wed Jul 16 07:12:33 2003
Last Write: Wed Jul 16 07:12:58 2003
Last Check: Mon Jul 14 15:08:08 2003
Unmounted properly
Last mounted on:
Operating System: Linux
Dynamic Structure
InCompat Features: Filetype,
Read Only Compat Features: Sparse Super,

META-DATA INFORMATION
--------------------------------------------
Inode Range: 1 - 184
Root Directory: 2

CONTENT-DATA INFORMATION
--------------------------------------------
Fragment Range: 0 - 1439
Block Size: 1024
Fragment Size: 1024

…
```

We see that the image was last mounted on 16th July 2003 and that the floppy ext2 partition had a block size of 1kb.

The file/directory browsing capability of Autopsy allows the user to navigate through the reconstructed filesystem and view and export the raw contents of files including deleted files. The files are appended with the extension `.raw`. The following files were exported and the `.raw` extension removed for viewing in a safe VM:

```
/John/sectors.gif
/John/sect-num.gif
/May03/ebay300.jpg
/nc-1.10-16.i386.rpm..rpm
/Docs/Mikemsg.doc
/Docs/Letter.doc
/Docs/Kernel-HOWTO-html.tar.gz
/Docs/DVD-Playing-HOWTO-html.tar
/Docs/Sound-HOWTO-html.tar.gz
/Docs/MP3-HOWTO-html.tar.gz
/.~5456g.tmp
```

The HOWTO files contained instructions on configuring Linux to play and support media playback including DVD support.

Using `file` on the hidden temp file `.~5456g.tmp` showed a `data` file type and loading the file into KHexEdit produces pure hexadecimal data with no human-readable strings.

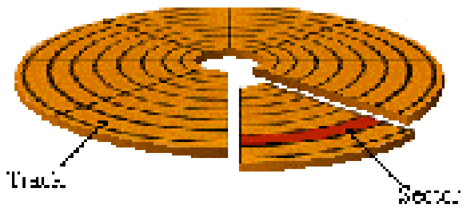
Figure 2 shows the reconstructed gif and jpg image files found on the floppy. According to Fig. 2a and 2b, the owner clearly, had an interest learning about the anatomy of hard drive tracks and sectors, consistent with the use of bmap. Fig. 2c shows an image to the ebay online auction website.

The Word document called `Mikemsg.doc` was viewed:

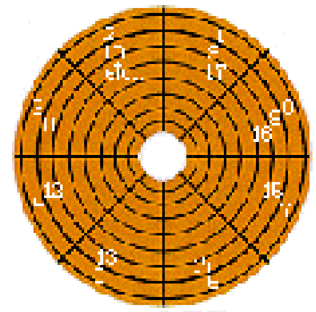
```
Hey Mike,

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.
```

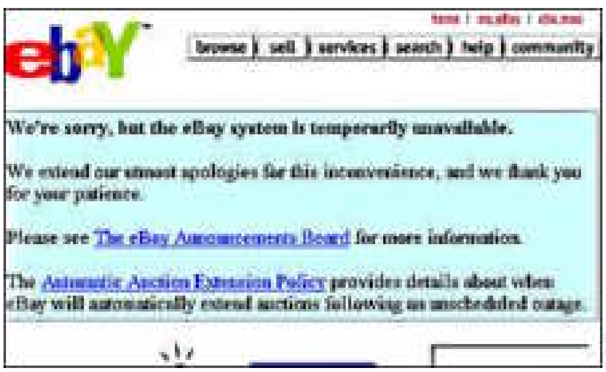
JP



(a)



(b)



(c)

Figure 2: Evidence images recovered from floppy drive left in suspect's drive: (a) illustration of hard disk track and sectors; (b) numerical layout of sectors ; (c) ebay Internet auction banner image

This is the one of the most important pieces of evidence found – a smoking gun. The letter talks about the acquisition of audio files and alludes to commercial sale (perhaps on ebay?) of these items as "advance orders". This would certainly be a main talking point if an interview was able to be held with the suspect. At this point, it unknown where the files were 'received' from.

Autopsy was then used to reconstruct the usage timeline. In Autopsy, the creation of a timeline uses several steps. Firstly, the powerful TASK `fls` command is used with the floppy image to output the files and directory structure contained within. Its power lies in that it can be used on 'live' or 'dead' systems and outputs both unallocated and deleted file information. Deleted files show '`(deleted)`' in the timeline. Using the `-r` switch without specifying a target inode allows the entire root directory structure to be processed recursively. Most importantly for timeline creation is the addition of the `-m /` switch tells fls to get the MAC time information and prepend the data with a '`/`'. Autopsy effectively runs the command:

 # fls -r -m / images/fl-160703-jp1.dd > /images/fl-160703.fls

The second step works at the Meta Data or inode layer in UNIX-speak. Inodes are the building blocks of the UNIX filesystem and are central to forensics. In the timeline reconstruction, we are also interested in 'catching' files that do not have an associated filename after deletion. In UNIX these deleted files still update their MAC times and this information can be extract from the inode data. Enter the `ils` tool that outputs deleted file inode information using the `-m` switch. We explicitly specify the filesystem using `-f` switch. Usage:

 # ils -m -f linux-ext2 images/fl-160703-jp1.dd > fl-160703-jp1.ils

The output from `ils` on, for example, a deleted file at inode 98 will appear in the timeline file as `<fl-160703-jp1-dead-98>`.

The two `ils` and `fls` are concatenated automatically by Autopsy into a single `body` text file.
The final step is uses the TCT/TASK `mactime` tool on `body` to output a chronological timeline in the format:
date,time/MACtime/type-permission/owners/inode#/name. The entries are grouped by common time.
The resulting timeline is best viewed using a text reader like `vi`.

The entire timeline file for the `fl0160703-jp1.dd` image is:

```
Tue Jan 28 2003 15:56:00    20680 ma. -/-rwxr-xr-x 502      502      25
/John/sectors.gif
                            19088 ma. -/-rwxr-xr-x 502      502      24
/John/sect-num.gif
Mon Feb 03 2003 11:08:00     1024 m.. d/drwxr-xr-x 502      502      12
/John
Sat May 03 2003 10:10:00     1024 m.. d/drwxr-xr-x 502      502      14
/May03
Wed May 21 2003 10:09:00    27430 ma. -/-rwxr-xr-x 502      502      19
/Docs/Kernel-HOWTO-html.tar.gz
                            29184 ma. -/-rwxr-xr-x 502      502      13
/Docs/DVD-Playing-HOWTO-html.tar
Wed May 21 2003 10:12:00    32661 ma. -/-rwxr-xr-x 502      502      20
/Docs/MP3-HOWTO-html.tar.gz
Wed Jun 11 2003 13:09:00    29696 ma. -/-rw------- 502      502      16
/Docs/Letter.doc
Mon Jul 14 2003 14:08:09        0 mac ---------- 0        0        1
<fl-160703-jp1.dd-alive-1>
```

```
                            12288 m.c d/drwx------ 0        0        11
/lost+found
Mon Jul 14 2003 14:11:50    26843 ma. -/-rwxr-xr-x 502      502      21
/Docs/Sound-HOWTO-html.tar.gz
Mon Jul 14 2003 14:12:02    56950 ma. -/-rwxr-xr-x 502      502      22
/nc-1.10-16.i386.rpm..rpm
Mon Jul 14 2003 14:12:15   100430 ma. -rwxr-xr-x 0        0        23
<fl-160703-jp1.dd-dead-23>
Mon Jul 14 2003 14:12:48    13487 ma. -/-rwxr-xr-x 502      502      26
/May03/ebay300.jpg
Mon Jul 14 2003 14:13:13   546116 m.. -rwxr-xr-x 502      502      27
<fl-160703-jp1.dd-dead-27>
Mon Jul 14 2003 14:13:52     2592 m.c -/-rw-r--r-- 0        0        28
/.~5456g.tmp
Mon Jul 14 2003 14:19:13   100430 ..c -rwxr-xr-x 0        0        23
<fl-160703-jp1.dd-dead-23>
Mon Jul 14 2003 14:22:36     1024 m.. d/drwxr-xr-x 502      502      15
/Docs
Mon Jul 14 2003 14:24:00   487476 m.. -/-rwxr-xr-x 502      502      18
/prog
Mon Jul 14 2003 14:43:44    26843 ..c -/-rwxr-xr-x 502      502      21
/Docs/Sound-HOWTO-html.tar.gz
                             1024 ..c d/drwxr-xr-x 502      502      15
/Docs
Mon Jul 14 2003 14:43:53    13487 ..c -/-rwxr-xr-x 502      502      26
/May03/ebay300.jpg
Mon Jul 14 2003 14:43:57    56950 ..c -/-rwxr-xr-x 502      502      22
/nc-1.10-16.i386.rpm..rpm
Mon Jul 14 2003 14:45:48    29184 ..c -/-rwxr-xr-x 502      502      13
/Docs/DVD-Playing-HOWTO-html.tar
Mon Jul 14 2003 14:46:00    27430 ..c -/-rwxr-xr-x 502      502      19
/Docs/Kernel-HOWTO-html.tar.gz
Mon Jul 14 2003 14:46:07    32661 ..c -/-rwxr-xr-x 502      502      20
/Docs/MP3-HOWTO-html.tar.gz
Mon Jul 14 2003 14:47:10   546116 .a. -rwxr-xr-x 502      502      27
<fl-160703-jp1.dd-dead-27>
Mon Jul 14 2003 14:47:57    29696 ..c -/-rw------- 502      502      16
/Docs/Letter.doc
Mon Jul 14 2003 14:48:15    19456 mac -/-rw------- 502      502      17
/Docs/Mikemsg.doc
Mon Jul 14 2003 14:48:53    20680 ..c -/-rwxr-xr-x 502      502      25
/John/sectors.gif
                            19088 ..c -/-rwxr-xr-x 502      502      24
/John/sect-num.gif
Mon Jul 14 2003 14:49:25     1024 ..c d/drwxr-xr-x 502      502      12
/John
Mon Jul 14 2003 14:50:15     1024 ..c d/drwxr-xr-x 502      502      14
/May03
Wed Jul 16 2003 06:03:00   546116 ..c -rwxr-xr-x 502      502      27
<fl-160703-jp1.dd-dead-27>
Wed Jul 16 2003 06:03:13     1024 m.c -/drwxr-xr-x 0        0        2
/John/ (deleted-realloc)
Wed Jul 16 2003 06:05:33   487476 ..c -/-rwxr-xr-x 502      502      18
/prog
Wed Jul 16 2003 06:06:15    12288 .a. d/drwx------ 0        0        11
/lost+found
```

```
Wed Jul 16 2003 06:09:35     1024 .a. d/drwxr-xr-x 502      502      12
/John
Wed Jul 16 2003 06:09:49     1024 .a. d/drwxr-xr-x 502      502      14
/May03
Wed Jul 16 2003 06:10:01     1024 .a. d/drwxr-xr-x 502      502      15
/Docs
Wed Jul 16 2003 06:11:36     2592 .a. -/-rw-r--r-- 0        0        28
/.~5456g.tmp
Wed Jul 16 2003 06:12:39     1024 .a. -/drwxr-xr-x 0        0        2
/John/ (deleted-realloc)
Wed Jul 16 2003 06:12:45   487476 .a. -/-rwxr-xr-x 502      502      18
/prog
```

Using the chronological timeline, allows an insight into the suspect's activities. The apparent research phase into hard disk sectors and tracks began early in Jan and February while the May's activities show an active interest in researching HOWTO files for setup DVD and media player support in Linux and in ebay online auction site. Later, in June and July, the real fun begins with the bulk of the suspect's activity. The creation of the letter to 'Mike' indicates receiving of media files prior to 14th July. Around 30 minutes before writing the letter, the netcat rpm was accessed possibly indicating network activity at around 14:12 on Mon July 14 2003.

Using bmap/prog on all the evidence files is a logical next step to check and see if any files contain hidden data in the slack space.

Using **`# bmap –checkslack <filename>`** shows whether a file contains data in the slack space. Running bmap on a directory produces the output '`<Directory> is not a regular file`'.
Bmap was executed on all evidence files as follows:

```
# bmap --checkslack nc-1.10-16.i386.rpm..rpm
nc-1.10-16.i386.rpm..rpm does not have slack
# bmap --checkslack prog
prog does not have slack
# bmap --checkslack DVD-Playing-HOWTO-html.tar
DVD-Playing-HOWTO-html.tar does not have slack
# bmap --checkslack Kernel-HOWTO-html.tar.gz
Kernel-HOWTO-html.tar.gz does not have slack
# bmap --checkslack MP3-HOWTO-html.tar.gz
MP3-HOWTO-html.tar.gz does not have slack
# bmap --checkslack Sound-HOWTO-html.tar.gz
Sound-HOWTO-html.tar.gz has slack
# bmap --checkslack Letter.doc
Letter.doc does not have slack
# bmap --checkslack Mikemsg.doc
Mikemsg.doc does not have slack
# bmap --checkslack sect-num.gif
sect-num.gif does not have slack
# bmap --checkslack sectors.gif
sectors.gif does not have slack
```

```
# bmap --checkslack ebay300.jpg
ebay300.jpg does not have slack
```

Clearly, the file `Sound-HOWTO-html.tar.gz` is worth further investigation. The number of hidden slack bytes is given by:

```
#  bmap --slackbytes Sound-HOWTO-html.tar.gz
805
```

The slack space data was saved to the normal filesystem with name `ebslack` using:

```
# bmap --slack --outfile /GCFA/ebslack Sound-HOWTO-html.tar.gz
getting from block 190
file size was: 26843
slack size: 805
block size: 1024
```

To find out what type of file it contains:

```
# file ebslack
ebslack: gzip compressed data, was "downloads", from Unix
```

Very interesting! This is a gzip compressed file that was a folder called `downloads`. Renaming this file with the proper extension

```
#  cp ebslack ebslack.gz
```

Then uncompressing the file using `gunzip`:

```
#  gunzip ebslack.gz
```

yields the file `ebslack` of type:

```
#  file ebslack
ebslack: ASCII text
```

Excellent – we can view this using `vi`:

```
# vi ebslack

Ripped MP3s - latest releases:
www.fileshares.org/
www.convenience-city.net/main/pub/index.htm
emmpeethrees.com/hidden/index.htm
ripped.net/down/secret.htm
***NOT FOR DISTRIBUTION***
```

We have a second smoking gun! The letter, interest in Linux media files and timeline activity makes sense now. The suspect appears to have been accessing illegal public file sharing websites to download newly released ripped off copies of Music CD's in the MP3 file format. The URL evidence implies that the suspect was interested in MP3 audio file sharing and downloading. Netcat may have been used to store downloaded MP3 files somewhere else on the corporate network thus pushing the blame elsewhere.

Legal implications
The actual acquisition and possession of the bmap program is certainly not illegal by any means and no UK laws are violated directly. However, the execution of the program in order to deliberately change data held on a computer may constitute a criminal offence under the UK parliamentary **Computer Misuse Act (CMA) 1990 section 3** [CMA].

Section 3-1(a) of the CMA indicates that a person is guilty of an offence when there is "*unauthorised modification of computer material*". More specifically, an offence is committed when a person deliberately performs "*any act which causes an unauthorised modification of the contents of any computer"* and to "*to prevent or hinder access to any program or data held in any computer; or the reliability of any such data.*" (Sec 3-2(b),(c)). The act is broadly applicable and certainly the use of data hiding tools would constitute unauthorised modification, prevent access to data held and potentially alter the reliability.

If successfully prosecuted under the CMA, the suspect would face a minimum prison sentence of six months and a maximum of five years or a fine or both.   Generally, a computer misuse allegation is a precursor to a more conventional crime like fraud or theft and the authorities have been more historically comfortable with prosecuting these conventional crimes.
To date, there is no UK case law precedence regarding the use of steganographic or forensic tools used to hide data that can be referenced. With anticipated increased use of these tools by cyber criminals and terrorists, it is a matter of time before they enter the court of law.

Considering now the actual data hidden. In this case, the forensic analysis revealed that was used to hide activities relating to the download of MP3 audio files on public servers. Additionally, a letter was found that pointed to the subsequent sale of these files on disk. These would certainly be criminal activities under UK law and are discussed in detail in Part 3 of this report.

With the floppy image evidence analysed, it cannot be absolutely proven that the bmap was run on the corporate network. Although the evidence suggests this *is* the case, to unequivocally prove whether the tool had been used on the corporate network, additional investigations are needed by the system administrators.

The acquisition of the bmap tool and it's compilation and execution would most definitely violate the corporate computer/network acceptable use policy. The purpose of the policy is to "*outline the acceptable use of computer equipment, in order to protect the Company and its employees. Inappropriate use potentially exposes the Company to the risks of virus/worm attacks, compromise of network facilities and services.*".

The coverage of the policy are defined in the overview statement:

"*All computer-related equipment including hardware, software, operating systems, storage media, network accounts, internet/intranet access and electronic mail are the property of the Company. As such, these systems are to be used for business purposes only i.e. those that serve the needs of the business, our clients and customers. It is a requirement that all employees know and adhere to these guidelines. The policy applies to all employees, contractors and other workers at the Company.*"

The Internet download acceptable use policy details that "*only material from bona-fide business, commercial or government websites may be downloaded, provided that the nature of the material downloaded is essential to the Company's business needs. No downloads for personal use will be tolerated.*"

The bmap tool hides data within the slack space and performs low level disk operations would certainly not constitute normal legitimate business needs and would clearly be a violation of the policy, leading to action by the Company.

Additional legal protection is offered by warning banners on computer systems advising that:

- "*access to the system is limited to Company authorise activity*"
- "*any attempted or unauthorised access, use or modification is prohibited*"
- "*use of the Company systems may be monitored and recorded*"
- "*if monitoring reveals any criminal activity, the company can provides the records to law enforcement*"

The opportunity to directly interview the suspect provides a golden opportunity to potentially prove whether the he did install and execute the binary on the company system deliberately. Good interview practice and questions may make or break this success.
In order to understand the artist, you have to understand the artwork. The same applies to interviewing suspects. You must understand the interviewee's motivations and develop a rapport. This is central to a successful interview. In achieving this, the interviewer will show an understanding of the suspect's world, casting aside any personal feelings regardless of the nature of the suspect's activities. The interview would be held away from the subject's workplace to facilitate this.

The interview questions
Clearly the subject of the investigation has good IT and Linux skills and a keen interest in obtaining illegal media files. A technical interview will ascertain both the extent of his knowledge and his activities. Insider threats are very common particularly in unhappy employees who may have a grudge against the boss/Company. These threats can be initiated by social issues such as the funding of an addiction, support, to supplement income and may reflect poor financial situation.

In addition it is worth considering whether It is not impossible that John Price was being blackmailed by an outsider to perform criminal activities using the corporate network.

Perhaps his “friend” Mike is blackmailing him? This is a path worth investigating. A personal interview will shed light on these issues.

I would ask the following questions to find out about John Price’s activities and lifestyle with an firm though friendly and professional manner.

Personal questions

What is the boss like to work for – he’s regarded as a tough taskmaster isn’t he?
What is your relationship with Mike?
What else can you tell me about Mike?
What’s your financial situation like?
If you could make some extra money selling audio material would you do so?

Technical questions

How would you describe your normal day-to-day duties at the Company?
How would you rate your knowledge of IT systems generally?
Are you the registered user of the account 502 on the system?
Have you been aware of anyone else accessing your account?
Were you logged on to this account and what were you doing on the system on that day?
When did you log out from your account
Have you ever heard of bmap?
Why did you wipe the hard drive of your computer?
Do you ever download music from the internet or have used a peer-to-peer sharing programs like Kazaa?
Have you ever used the program netcat at work?
Did you execute the file called prog on the system?
Do you ever sale anything on ebay – it’s a great site?

Hopefully the above set of questions would be enough to convince John Price to confess to owning and executing prog on the system with the intent of sharing files across the corporate network. If not, more investigations by the system administrators are required.

Case Information

A computer forensic investigation of the floppy image acquired from the Company revealed significant evidence regarding the activities of suspended employee John Price. The following case summary information would be presented to the Company’s system administrators to help them detect whether the binary is still in use or has been used on any other systems. A case summary of the steps taken in the forensic analysis and the corresponding evidence recovered is given in Table 3.

In summary, detailed computer forensic methods were applied to the floppy image acquired from that Company that revealed the nature of an unknown binary called prog. The file was shown to be identical to a tool called bmap that, when executed on target file, allows the user stuff hidden data into a file’s slack space. Slack space is normally unseen by the system and is the residual space between data and the end of the file storage block allocated on the disk. This hidden data, rather like invisible ink, would remain invisible until the tool is reapplied to the same file thereby revealing the hidden message. The tool and several other evidence files were noted to belong to the user with uid and gid 502 on the Linux system and are indicated to be the suspect John Price.

Additional files were found on the reconstructed floppy image. All files on the disk were checked for any covert data hiding in the slack space made indicating use of the tool. One of the files, `Sound-HOWTO-html.tar.gz`, revealed a compressed file within the slack space that, when uncompressed, contained an important text message detailing several websites where illegal 'ripped' copies of commercial music CD's are publicly shared. This is indicative that John Price may have been using the corporate network to distribute copyrighted material. Further evidence of this are a letter found that details the acquisition of new audio files and advanced orders relating to their potential sale.

Further analysis showed that VMWare virtualisation software may have been used, perhaps to aid hiding of his activities. The software could be run in the background of normal work activities while files downloaded were progressing.

The netcat client-server tool was also found indicating that potentially the downloads could have been performed onto another corporate machine. Netcat can be used to set a server to listen at any port number using the TCP or UDP protocols [NC]. This would allow the suspect to setup a tunnel across the corporate network to potentially send the file shares out to "Mike" external to the company. To subvert the border firewall, for example, netcat could be configured to send/receive out on the DNS port (UDP 53).

It is prudent that the system administrators check the usage of the corporate network looking particular for MP3 or media file downloads in network traffic logs and also check the firewall logs for suspicious activity. Certainly webserver logging should be active and details noted any IP addresses performing HTTP file uploads/downloads requests to the following URLs:

```
www.fileshares.org/
www.convenience-city.net/main/pub/index.htm
emmpeethrees.com/hidden/index.htm
ripped.net/down/secret.htm
```

From the letter to Mike it is evident that illegal media files were downloaded at night – perhaps the scheduling `cron` command was used to automatically initiate downloads or the suspect worked late at night while the office was empty. Checking the system to see what users were logged on, particular user 502, were logged on out of normal office hours may help detect this misuse of the corporate network.

The analysis also showed that the suspect had a strong interest in researching low level disk sector information (including disk sectors and tracks) in order to understand how the bmap tool worked. This research phase of the suspect's activities were noted to occur from a forensic timeline analysis in the period Jan – July 2003. These included web documents downloaded that detail HOWTO's in configuring Linux to allow media playing and adding DVD support. Additionally, the actual bmap tool would have downloaded at some point, most likely in July possibly via the corporate network.

Analysis step	Evidence found
Initial examination of image using `file` command. Mounting of the image onto the local Linux filesystem	Original disk was a Linux ext2 filesystem. Revealed of original file and directory structure
Analysis of prog binary attributes using `debugfs`, `ls`	prog is owned by userid 502, is 487Kb in size was last accessed on 07/16/03 and last modified on 07/14/03
Extraction of prog strings using `strings` command.	Associated keywords found including "newt,wipe,place,bmap,slack" were found
Post-mortem binary analysis of prog	Prog is a statically-linked, stripped binary Linux binary
Live behavioural analysis of prog inside protected virtual machine lab running snort network sniffer	No network traffic associated with program. Discovered nature of prog as a program which shows low-level disk information file parameters such as fragmentation and slack space performs fundamental disk operations
Google Internet searches on prog keywords. Dowloading of bmap tool source code and initial analysis	Identification of a data hiding tool called bmap that sounds like prog. Reveal bmap is a Linux block mapping utility that performs low level disk operations and can hide data within file slack space
Comparison of bmap and prog	The two programs are identical in behaviour.
Building of a stripped, statically-linked bmap binary to match the found prog	File sizes and hashes of bmap and prog do not match due to editing of the prog source code, build parameters
Detailing the forensic footprints left through program building to execution	The building, installation and testing will produce footprints in `.bash_history`, in webserver logs and in file MAC times. The execution of bmap leaves an unobvious footprint and no network processes are involved
Autopsy analysis of floppy filesystem.	Reconstruction of filesystem usage timeline showing suspect's activities between 28 Jan and 16 July 2003
Examination of other evidence found on floppy disk	Images found describing hard disk sectors & tracks, severeal documents detailing how to setup Linux to play DVD's and MP3 audio. A document was found detailing the acquisition of media files with intent to sell them on
Execution of bmap on the evidence to check for hidden messages	One document, `Sound-HOWTO-html.tar.gz` contained a compressed,gzip file. The uncompressed file showed text containing specific file sharing websites where ripped MP3 of commercial music can be downloaded

Table 3: Step-by-step summary of analysis steps taken on floppy image and evidence found

Bmap is a relatively obscure tool and it is highly likely that it was either downloaded from the Scyld Corporation website of from Packetstorm security [SCY,PSS]. Any web visits to these websites should be treated with extreme suspicion and downloads may be observed in webserver logs. The downloads would either be in the form in HTTP or FTP requests.

It is highly likely that this would be accompanied by web search queries were performed on Linux data hiding tools and slack space. These activities are certainly not normal business usage and would be easily spotted in the HTTP webserver logs. Backtracking logs to a specific IP address would show which corporate machine (if any) was responsible for the downloads. However, it is possible that the tool was downloaded off-site, maybe at the suspect's home.

Detecting the actual execution and use of the tool would be very difficult. This is largely due to the fact it is small enough to be run from a floppy and the forensic analysis showed that the tool runs transiently on a target file, leaving a minimal forensic audit trail. Additionally, there is no signature network traffic associated with tool execution that would indicate exactly where and when it had been used. Clearly it would be very laborious to image all of the hard drives on the corporate machines and perform `string` keyword searches. If absolute detection was sought this would be tedious but possible.

The tool would need to be built from source code prior to use. The necessary creation of a stripped, static binary would be reflected in access times of the `Makefile` and source code files. If a standard bmap install was used then it would appear in the `/usr/local/bin` directory of the system.
Straightforward execution of the `find` command on each system looking for prog and bmap might produce a lucky strike that find the program directly. The execution, compilation and installation steps on a UNIX system would likely leave an audit trail in the user's `.bash_history` file. This file is often overlooked but contains important command history evidence that would show usage on other machines.

Additional information
Hashing Algorithms
Bmap: Linux Data Hiding and Recovery
Netcat
UK Computer Misuse Act 1990

4. Case #2- Trash and Treasure: Computer Forensics and Public Domain Data

The primary aim of this case study was to increase awareness of the type of evidence trail that is left in the public domain where computer systems are left for trash.

Performing computer forensic techniques on a public domain system is comparatively difficult to a known system. Firstly, nothing is known about the state of the system or previous users at the time of acquisition. It is unknown whether the hard drive is physically readable, has been wiped or indeed exists at all! Moreover, there is no prior knowledge of the system history or if it has been compromised in any way.

A detailed forensic analysis on a real world system was performed to help answer the following questions:

- What was the usage of the system, what operating systems, software and hardware were installed?
- What type of data exists on the systems (if any)?
- Has the computer been compromised?
- Can we recover any deleted files?
- What were the habits of the owner?

Synopsis of case facts
The author acquired systems using various methods, including online auctions, computer vendors and entire computer systems left at public refuse dumps to be trashed. The case study described herein was obtained using the latter method but is representative of public systems obtained by any of the above means. Refuse dumps provided the easiest and cheapest means of obtaining systems.

In the UK, most recycling centres are managed by private companies that can sell items to the public at very low cost. Typically, a complete box can be bought for ca. £3 (~$1.80).
Data is a waste by-product of the digital age and 'dumpster diving' is the past-time of searching through this public 'waste' for goodies and information [DD]. Dumpster diving has provided computer hackers with a gold mine of IT-related information when 'casing a joint'.

Searching trash for treasure is a powerful tactic since it exploits a fundamental psychology installed in all human beings that *trash is bad:* once something is condemned to trash, we prefer to forget about it and place no emotional or monetary value on the forgotten item.

Disclaimer
The author bought the systems completely legally for the purposes of collecting spare PC parts and to perform this research. The intention is to increase awareness of the dangers of 'trashing' computer systems in order that we can further secure our data. No data subsequently found was kept or used in any way. Any personal information has been sanitised herein and after completion of this study, all media were either destroyed or securely wiped.

Description of system under analysis
The author bought a completely integral used Viglen brand PC box from a public recycling centre for £3. At the time of acquisition, nothing was known about what it had been used for, or the system/OS configuration contained within. Moreover, it was not

known whether the hard drive had been removed or not. The author had often used Viglen brand PC's in the period 1990-1995 and knew that, at this time, they were the PC of choice for small businesses and universities. This provided a lead that, if the hard drive still existed, it may contain sensitive corporate data. At the time of acquisition, it was noted that the system was in immaculate physical condition and there were no signs such as dents or screwdriver marks indicating access to the internals of the box.

Table 4 shows the seizure details of case #2 study reported. The system was found at a public refuse dump, left open for public sale or salvage.

Hardware
All of the following items in Table 1 were seized from a UK public refuse tip and tagged according to ID #: `'investigator initials'-mmyyx`. The hard drive was carefully removed using an anti-static wrist strap and screwdriver and placed inside a tagged plastic evidence.

Tag ID# : investigator	Description	Notes
#MAS-0803A : Michael Scott	Viglen Contender Pro 4DX266 desktop PC Serial #: 4D60706 250MB internal hard drive, 3.5" floppy drive, internal sound card, serial / parallel ports.	Box in exceptional condition despite age (ca. 1993).
#MAS-0803B : Michael Scott	Western Digital Caviar 2450, 540.8MB Hard Drive S/N #: WT261 122 7036, 1048 cylinders	Jumper was set on Master when seized.

Table 4: Evidence tag ID's and description of items seized

Image Media
In Red Hat Linux, IDE hard drives are designated with device labels `hd`, as follows:

`/dev/hdd` Master drive on primary IDE connection
`/dev/hdb` Slave drive on primary IDE connection
`/dev/hdc` Master drive on secondary IDE connection
`/dev/hdd` Slave drive on secondary IDE connection

These are appended with 1,2,3.. to represent the first, second and third etc., partitions. Thus `/dev/hdd1` is the first partition on disk `/dev/hdd1`.

Prior to imaging, the existing master IDE CD-ROM on the secondary connector was removed to avoid any potential conflicts between the master and slave drives noted by Dittrich [DIT]. The evidence disk was then switched to "slave" mode by switching the hardware jumper and then connected to the secondary IDE motherboard connection, `/dev/hdd`. In this way, we avoid accidentally mounting the evidence drive, corrupting the evidence and destroying its admissibility in court.

The computer BIOS (basic input/output system) was entered and checked that the evidence drive was not tagged as bootable and that the system recognised its existence. The computer was then booted into Linux rather than Windows which would automatically mount the FAT drive.

Once Linux had booted, running the `fdisk` command in interactive mode using the `-l` switch shows the partition table. The partition table allows us to see all the disk partitions that exist (if any) on the evidence drive and is very useful as it tells us the filesystem type, typically Linux ext2, FAT16/32 or NTFS. Usage:

```
# fdisk -l
Disk /dev/hdd: 540 MB, 540868608 bytes
32 heads, 63 sectors/track, 524 cylinders
Units = cylinders of 2016 * 512 = 1032192 bytes
Device     Boot      Start      End     Blocks   Id   System
/dev/hdd1    *          1       523     527152+  6    FAT16
```

showing the hard drive parameters and that `/dev/hdd1` is a DOS/Windows FAT16 partition.
An md5 hash of the actual device and device partitions were taken and saved to files for comparison:

```
# md5sum /dev/hdd > cav.hdd.md5
533f6d959c8c902cfffda538e50c2ed0             /dev/hdd
# md5sum /dev/hdd1 > cav.hdd1.md5
a946683a26baf79ceaec662dab9ec6f6             /dev/hdd1
```

Prior to drive imaging, a forensic images directory was created:

```
# mkdir  /mnt/images
```

The previously sterilised `/dev/hda5` partition where the disk images are held was mounted onto the directory by executing:

```
# mount   /dev/hda5  /mnt/images
```

The `dd` command was used in Linux to produce an bit-by-bit copies of the input drive and partition as follows

```
#  dd  if=/dev/hdd  of=/mnt/images/cav.hdd.0803.dd
#  dd  if=/dev/hdd1 of=/mnt/images/cav.hdd1.0801.dd
```

no errors were output, indicating no bad disk sectors or disk read problems.

The machine was powered down and the drive was then removed from the system and placed back in the evidence bag.

The image md5 hashes of the drive and partition images were taken:

```
# md5sum    /mnt/images/cav.hdd.110803.dd
533f6d959c8c902cfffda538e50c2ed0 /mnt/images/cav.hdd.110803.dd
# md5sum   /mnt/images/cav.hdd1.110803.dd
a946683a26baf79ceaec662dab9ec6f6  /mnt/images/cav.hdd1.110803.dd
```

Clearly the hash values of the originals and images are identical, proving that the imaging process has maintained the integrity of the evidence.

Media Analysis of System

<u>Reconstruction of original filesystem</u>

The partition image, `cav.hdd1.110803.dd` was used to analyse the filesystem. The mount point directory for the forensic image was created using **`# mkdir /mnt/winanalysis`**

The FAT16 filesystem (`vfat`) image was mounted onto the Linux `ext2` mount point `/mnt/winanalysis` using the loopback option again and protecting the mounted image as follows:

```
# mount -t vfat –o ro,loop,noatime,noexec,nodev
cav.hdd1.110803.dd  /mnt/winanalysis
```

Performing **`# ls`** shows the reconstructed windows root file structure:

```
apology autoexec.syd config.sys    drvspace.bin  msworks      wina20.386
autoexec.bak  clutil     corel40       io.sys        pkunzip.exe windows
autoexec.bat  command.com   ~doc0b19.tmp  legato     pm3
winword  autoexec.cor    config   dos    msdos.sys   utils      word6
```

Checking that there the filesystem does not allow write access, I used `touch` to try and create a new file:

```
# touch mas
touch: creating 'mas': Read-only filesystem
```

Now, using Linux we are free to browse through this filesystem and view files without worrying about modifying them in the process. Any deleted files need to be analysed using specialist forensic tools such as Autopsy or TCT.

Firstly, we want to enumerate the system and get a feel for the hardware and general setup at the time of usage.   The file `C:\config\4d60706.txt` was examined:

```
SYSTEM CONFIGURATION AND SPECIFICATION
Model: Contender Prof. 4DX266           Serial Number: 4D60706
SYSTEM PROCESSOR
```

```
     Processor: 80486                      Speed: 66MHz
     Coprocessor: 80x87                    No Weitek
SYSTEM MEMORY
     Base Memory: 640k bytes
     Shadow Memory: 384k bytes
     Extended Memory: 15360k bytes

FLOPPY DISK DRIVES
     Floppy Drive A: 1.44M bytes
     Floppy Drive B: None bytes

HARD DISK DRIVES
     Hard Drive 0:       Size:514M bytes
     Ha rd Drive 1: None
     Setup  Hard  Disc  0:    Type:40             Cyl:523     Hd:32
Sct:63

DISPLAY CONFIGURATION
     Display: Cirrus VGA
     Video Memory: 1024k bytes

I/O PORTS FITTED
COM1:3F8h      COM2:2F8h  LPT1:378h

This is the information about your system as it was shipped to you.
```

Here we have the entire factory configured hardware settings. The serial number matches that observed on the PC box at seizure time.

Furthermore, the file `C:\Windows\mouse.ini` showed that a standard two button PS-2 mouse was installed.

Numerous system files can provide important system information. One of these is `winsetup.inf`, the Microsoft windows setup file :

```
Defaults used in setting up and names of a few files
    startup   = WIN.COM
    defdir    = C:\WINDOWS
    shortname = Windows
    welcome   = "Windows 3.1"
    deflang   = enu
    defxlat   = 437
    defkeydll = usadll
    register  = "regedit /s /u setup.reg"
    tutor     = "wintutor.exe "
    NetSetup  = FALSE
    MouseDrv  = TRUE
```

```
    Version   = "3.1.040"

; Names of the disks Setup can prompt for.
[disks]
    1 =. ,"Microsoft Windows 3.1 Disk #1",disk1
    2 =. ,"Microsoft Windows 3.1 Disk #2",disk2
    3 =. ,"Microsoft Windows 3.1 Disk #3",disk3
    4 =. ,"Microsoft Windows 3.1 Disk #4",disk4
    5 =. ,"Microsoft Windows 3.1 Disk #5",disk5
    6 =. ,"Microsoft Windows 3.1 Disk #6",disk6
```

Showing that the operating system was Windows 3.1 version 3.1.040 and was installed into the directory `C:\WINDOWS`. Additional information given is that installation expected six floppy disks. The file `winver.exe` was executed on the protected system and showed the banner "`386 enhanced windows version 3.95 © 1991, Microsoft Corp`".

The Windows system initialization file, `C:\Windows\system.ini`, contains information that Windows and applications need to configure themselves including drivers and dynamic link library (DLL) files:

```
[boot]
shell=progman.exe
mouse.drv=C:\WINDOWS\mouse.drv
network.drv=
language.dll=langeng.dll
sound.drv=mmsound.drv
comm.drv=bicomm.drv
keyboard.drv=keyboard.drv
system.drv=system.drv
386grabber=AVGA.3GR
oemfonts.fon=VGAOEM.FON
286grabber=VGACOLOR.2GR
fixedfon.fon=VGAFIX.FON
fonts.fon=VGASYS.FON
display.drv=256_1280.DRV
drivers=mmsystem.dll
oldcomm.drv=comm.drv

[keyboard]
subtype=
type=4
keyboard.dll=kbduk.dll
oemansi.bin=

[boot.description]
keyboard.typ=Enhanced 101 or 102 key US and Non US keyboards
```

```
mouse.drv=Microsoft Mouse version 9.01
network.drv=No Network Installed
language.dll=English (International)
system.drv=MS-DOS System
codepage=437
woafont.fon=English (437)
aspect=100,96,96
display.drv=GD5426/28 v1.32B, 640x480x256

[386Enh]
32BitDiskAccess=OFF
device=*int13
mouse=*vmd
network=*dosnet,*vnetbios
ebios=*ebios
woafont=dosapp.fon
display=VDD542X.386
EGA80WOA.FON=EGA80WOA.FON
EGA40WOA.FON=EGA40WOA.FON
CGA80WOA.FON=CGA80WOA.FON
CGA40WOA.FON=CGA40WOA.FON
keyboard=C:\WINDOWS\mousevkd.386
device=vtdapi.386
device=*vpicd
device=vtda.386
…
FileSysChange=off
PagingFile=C:\WINDOWS\WIN386.SWP
MaxPagingFileSize=20460
device=C:\DOS\VFINTD.386
device=VSHARE.386
device=WDCDRV.386


[mci]
WaveAudio=mciwave.drv
Sequencer=mciseq.drv
CDAudio=mcicda.drv
CorelMOVE=C:\COREL40\PROGRAMS\mcicmv40.drv

[drivers]
timer=timer.drv
midimapper=midimap.drv
VIDC.MSVC=msvidc.drv…
```

The `win.ini` file contains settings about the Windows environment including user preferences, desktop preferences and logon information:

```
[windows]
…
spooler=yes
load=printman.exe
device=HP LaserJet 4L,HPPCL5E,LPT1:

[Desktop]
Pattern=(None)
Wallpaper=(None)
GridGranularity=0

[Extensions]
…
wps=C:\MSWORKS\msworks.exe ^.wps
reg=regedit.exe ^.reg
doc=C:\WINWORD\winword.exe ^.doc
lwp=C:\LEGATO\legato.exe ^.lwp

[intl]
sLanguage=eng
sCountry=United Kingdom
iCountry=44

[Microsoft Word 2.0]
INI-path=C:\WINWORD
programdir=C:\WINWORD

[MS User Info]
DefName=Authorized User
DefCompany=Viglen Ltd
…
```

It can be seen from the above files that the system had a HP LaserJet 4L printer installed on the Parallel Port (LPT1) and that no network was installed. The system was installed in the UK and it looks like the OS was pre-installed by the vendor Viglen Ltd. at the time of purchase since the user info showed '`Authorized User`' and '`Viglen Ltd`', respectively.
From the above details and the directory output, we see that the main programs installed were MSWorks, Microsoft Word 2.0, Word 6, CorelDraw 4, pkunzip and Legato. Legato is a collection of charting, database, spreadsheet and document applications and would be typical of business use in this era.

The directory `pm3` shown might as first glance look like Partition Magic but was actually the 'Premier Manager 3' soccer management game by Gremlin Interactive [PM3]!

The file `README.TXT` showed that MS-DOS v6.22 was installed along with Windows 3.1.

The `find` command was used to locate files containing the text `log` since log files often contain important information such as installation or virus scanner logs. Usage:

 # find /mnt/winanalysis –name '*log*'

This returned only the CorelDraw 4 install log file, `logfile.txt` that showed it was a full installation was successfully completed on 15th March 1995:

```
;;; Setup Log File Opened 03/15/95 14:25:20

Installing from floppy
Full Installation Selected
Base Directory: C:\COREL40…
Setup Succeeded
;;; Setup Log File Closed 03/15/95 14:52:51
```

The file `cirrus.inf` was viewed:

```
[data]
    Version   = "3.1"
[disks]
    1 =.,"Cirrus Logic GD542x Windows 3.1 Driver Diskette",disk1
[oemdisks]
    N =.,"Cirrus Logic GD542x Windows 3.1 Driver Diskette",vdd542x.386
```

showing that the installed graphics card contained a Cirrus Logic GD542x chipset and the driver is supplied on a single floppy as the file `vdd542x.386`. With combined information we know that it was a VESA local bus card with 1MB DRAM.

Word 6 setup `ini` file shows that installation of Winword occurs via nine floppy disks, namely:

```
[Source Media Descriptions]
"1","Microsoft Word: Disk 1 - Setup","ACMSETUP.EX_","."
"2","Microsoft Word: Disk 2","WINWORD.E1_","..\disk2"
"3","Microsoft Word: Disk 3","WINWORD.E2_","..\disk3"
"4","Microsoft Word: Disk 4","WINWORD.E3_","..\disk4"
"5","Microsoft Word: Disk 5","WINWORD.HL_","..\disk5"
"6","Microsoft Word: Disk 6","GR_BR.LE_","..\disk6"
"7","Microsoft Word: Disk 7","TABLES.DO_","..\disk7"
"8","Microsoft Word: Disk 8","CLASSIC1.WZ_","..\disk8"
"9","Microsoft Word: Disk 9","WORDCBT.LE_","..\disk9"
```

Now that the OS type is known and we have some initial data regarding the system configuration, Autopsy was used to dig further into the filesystem.
Autopsy was used as an analysis tool as it offers a fast and convenient method of mounting the disk image and allows extraction of all filesystem information from the

inode layer through to the ‘human-readable’ file data. Using the same method detailed in pg. 32, I setup a new Autopsy case with timezone set to GMT using the author’s initials “MAS” as the investigator and setting the mount point to `C:\`.
The FAT16 image, `cav.hdd1.110803.dd` was mounted by Autopsy automatically in read-only mode. All Autopsy case data was contained in the directory `/forensics/caviar/host`.

Clicking the image information button gives:

```
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT
OEM:MSDOS5.0
Volume ID: 543297758
Volume Label: MSDOS 6ù22
File System Type (super block): FAT16

META-DATA INFORMATION
--------------------------------------------
Range: 2 - 16864258
Root Directory: 2

CONTENT-DATA INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 16384
Sector of First Cluster: 291
Total Sector Range: 0 - 1054303
FAT 0 Range: 1 - 129
FAT 1 Range: 130 - 258
Data Area Sector Range: 259 – 1054303…
```

In comparison with the 1k block/cluster size of the Linux floppy in case#2, we see that this old FAT16 filesystem has very large large cluster sizes (16kb). This leads to large slack space wastage. In theory, if bmap was used on this drive significant amounts of data could be hidden into the file slack space!

To further parse data on the drive and ease analysis, the `sorter` command was executed to categorise files using their `file` description. The tool has the further advantage that it operates on both deleted and undeleted data.
Autopsy executes:

```
# sorter -h -m 'C:/' -d '/forensics//caviar/host/output/sorter-cav.hdd1.110803.dd/' -f fat16 -s '/forensics//caviar/host/images/cav.hdd1.110803.dd'
```

and produces a series of output directories according to file type. An extract is shown:

```
Results Summary
Images
/forensics//caviar/host/images/cav.hdd1.110803.dd
…
documents (128)
exec (655)
images (120) (thumbnails)
system (68)
text (177)
```

Each directory output from sorter has its own `index.html` file that allows browsing using mozilla or other web browser. The `images` directory was noted to contain only CorelDraw canvas and windows system bitmaps, as shown in the sample Fig. 3. The audio folder was observed to contain only standard Windows 3.1 sounds.

From the text directory, it is easy to browse system text files and several were exported from Autopsy for analysis. These are exported to a writeable directory on the media analysis workstation and then the `.raw` extension removed and replaced with `.txt`. The `images-cav.hdd1.110803.dd-C..WINDOWS.PROGMAN.INI.raw file` was shown to be the Windows Program manager initialisation file. Viewing this file shows the program manager groups, e.g. the startup group, `Group4=C:\WINDOWS\STARTUP.GRP`.

The file `printer.readme` shows:

```
README FILE FOR HEWLETT-PACKARD HPPCL5E PRINTER DRIVER WITH
SUPPORT FOR THE HP LaserJet 4/4M, 4L, 4ML, 4P/4MP, & 4Si/4Si MX
PRINTERS 31.V1.25 for Windows 3.1
```

Numerous other files in the `sorter` were examined but did not provide any significant extra information about simple system configuration.

Using a neat trick with the Linux Samba service, the Linux-mounted Windows FAT filesystem can be "exported" back into a real windows system for further analysis. This allows viewing of files in their native applications. For example, a word `doc` file can be viewed in Word 97 installed on the virtual machine. Sometimes file formats will be encountered that are not easily read outside their native applications, so this technique is useful.

The Samba share can be served on the same hardware using VMWare. This allows the host to share the mounted windows drive as a Samba share and the guest Windows VM to see this share. I had no Windows 3.1 operating system to run in VMWare so Windows 2000 Professional was used. Installed on the Windows 2000 VM were copies of Microsoft Office 97, Lotus Notes, McAfee Virus Scanner, Anti-trojan 5.5, WinAmp media player and a toolkit set of forensic tools.

To do this, on the workstation host `(Linuxanalysis)`, the Samba configuration file in `/etc/samba/smb.conf` was edited to include the windows 2000 VM as follows:

```
[winanalysis]
path = /mnt/winanalysis
guest ok = yes
writeable=no
```

specifying the previous mount path and ensuring read-only access.

VMWare was then executed using:

```
#   vmware  &
```

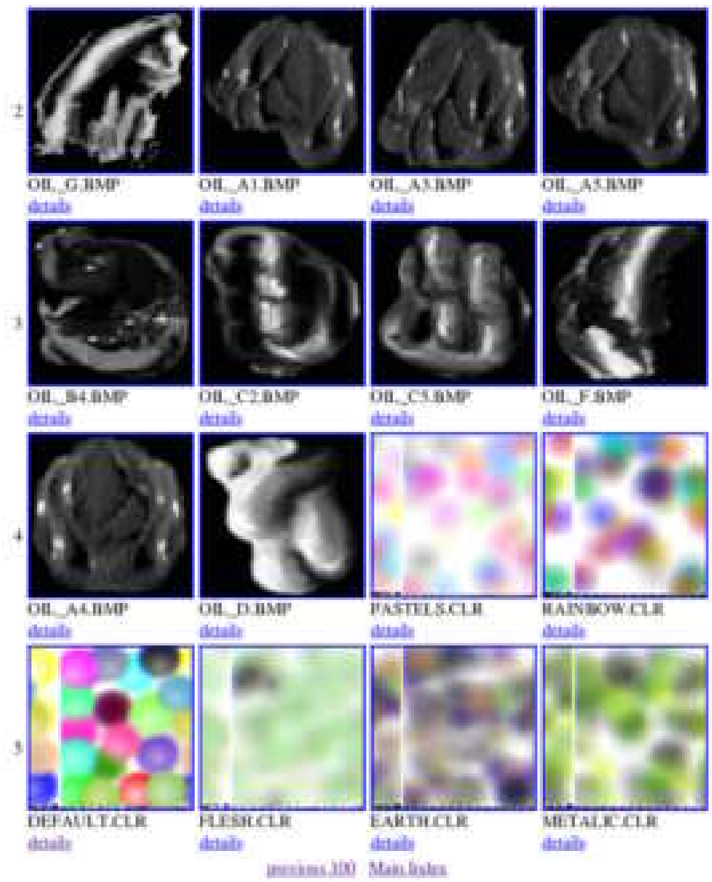


Figure 3: Example images output from sorter showing CorelDraw 4.0 Canvas bitmap files

and the `winanalysis` Windows 2000VM guest started. So that `winanalysis` could see the samba fileshare, the `net use` admin command was executed to map a network resource from the Linux samba share to the local Windows E: drive:

```
C:>  net use E: \\Linuxanalysis\winanalysis
```

The resultant directory/file structure is shown in Fig. 4a as seen in the Windows 2000 guest. The 540MB hard drive was just over 50% used: perhaps a lot of files had been deleted prior to "trashing". Note only undeleted files are seen in this case. Autopsy will be used later to recover deleted files.

Another great advantage of the combined workstation/Samba/VM technique is that further forensic tools, virus scanners and trojan horse scanners can be executed within the protected virtual machine.

McAfee Anti-virus V6 was executed on the E: drive. No viruses were found, as shown in Fig. 4b. In addition, Anti-Trojan 5.5 was executed and again no trojan horse infection was found.

After the analysis the disk image md5 hash was calculated using md5sum again:

```
# md5sum   /mnt/images/cav.hdd1.110803.dd
a946683a26baf79ceaec662dab9ec6f6  /mnt/images/cav.hdd1.110803.dd
```

we see that in our media analysis, the tools chosen did not alter the integrity of the evidence disk image.

Timeline Analysis

As in the case of the floppy image in case #1 described earlier, Autopsy was again used to create the filesystem timeline as it produces the timeline quickly and efficiently. As seen previously, the timeline provides valuable information in a case regarding suspect habits and gives important information about the system and installations via the MAC times.

A great feature of Autopsy in timeline analysis that is often unreported is that it will output a daily summary of activity in the filesystem timeline. This is a great help in examination as it shows "busy days" where major installations and updates were performed[11]. The PowerLoad DOS Timeline website gives a detailed history of Microsoft MS-DOS and Windows releases in the period 1980 - 2002, helping greatly in historic timeline analysis [PL1,PL2]. The file permissions and owners are redundant in the Windows system and are omitted for clarity. The salient events in the timeline are:

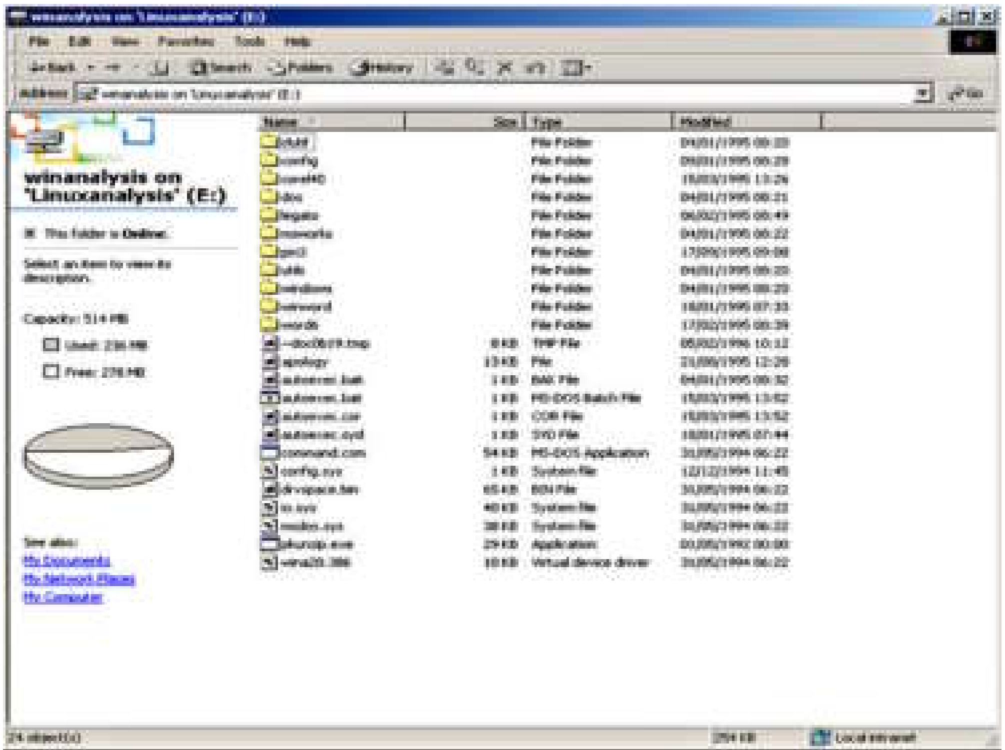
First activities occurred on May 22 1990: the physical format of disk using `physform.com` was followed by the installation of Cirrus logic video driver (VPDRV2_0.EXE) and disk parking by the vendor Viglen (DISKPARK.EXE).

The Windows 3.11 Operating system would have been pre-installed by the vendor, but the exact details of installation date cannot be determined from the timeline obtained.

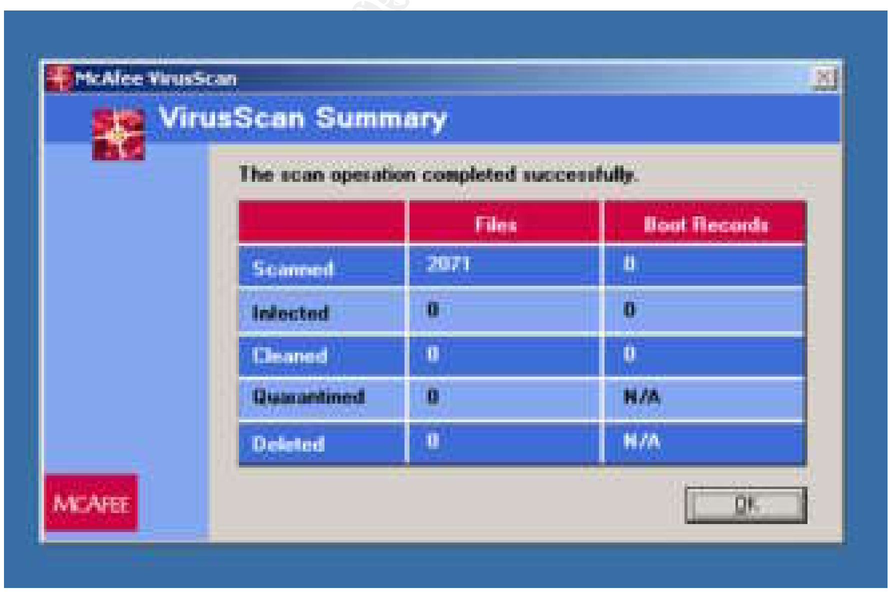
Tue May 22 1990:
```
5954      m..   324618     C:/UTILS/PHYSFORM.COM
```
Wed Sep 26 1990
```
13418     m..   398345     C:/UTILS/CIRRUS/GEM3VP3/VPDRV2_0.EXE
```
Fri May 03 1991
```
5664      m..   324615     C:/UTILS/DISKPARK.EXE
```
…

[11] Using a spreadsheet e.g., Excel, my strategy for analysing key timeline events is to import the `timeline.sum` summary file output from Autopsy into 'date' and 'activity count' columns and sort them in ascending order. This way you can clearly see the busiest days. The timeline is then used to correlate the events by particular dates.

(a)



(b)



<u>Figure 4</u>: Screen captures of: (a) reconstructed window filesystem using Samba and VMWare; (b) McAfee virus scan of hard drive in Case #2 showing no viruses

Fri Jun 28 1991: Legato map clipart installed

```
9650        m..   2072152    C:/LEGATO/DENMARK.WMF
5904        m..   2072149    C:/LEGATO/AUSTROTL.WMF
6566        m..   2072153    C:/LEGATO/FRANCE.WMF
742         m..   2072165    C:/LEGATO/USAOUTL.WMF10646          m..
      2072163     C:/LEGATO/SWTZRLND.WMF
…
```

Thu Nov 19 1992: Word for Windows 2.0 installed

```
1278240     m..   91665      C:/WINWORD/WINWORD.EXE
195952      m..   91653      C:/WINWORD/SETUP.EXE
78336       m..   91694      C:/WINWORD/WRITWIN.CNV
550400      m..   107525     C:/WINDOWS/MSAPPS/MSGRAPH/GRAPH.EXE
71984       m..   1770501    C:/WINDOWS/MSAPPS/GRPHFLT/EPSIMP.FLT
…
```

Sun Feb 14 1993: HP Printer Drivers installed

```
89292       m..   425653     C:/WINDOWS/SYSTEM/HPPCL5C7.DLL
18480       m..   425650     C:/WINDOWS/SYSTEM/HPPCL5C4.DLL
22496       m..   425651     C:/WINDOWS/SYSTEM/HPPCL5C5.DLL
408864      m..   425648     C:/WINDOWS/SYSTEM/HPPCL5C2.DLL
…
```

Tue Jul 20 1993: Corel Draw 4.0 installed

```
25088       m..   2547250    C:/COREL40/PROGRAMS/CCAPTURE.EXE
48096       m..   2547289    C:/COREL40/PROGRAMS/W4W05F.DLL
694810      m..   2547276    C:/COREL40/PROGRAMS/CTRTIGER.DLL
14264       m..   2609201    C:/COREL40/CUSTOM/CNSTRUC1.PAT
1998912     m..   2547224    C:/COREL40/PROGRAMS/CORELDRW.EXE
…
```

Mon Aug 16 1993: Mouse Installed and configured

```
10432       m..   311938     C:/WINDOWS/POINTER.EXE
2742        m..   324619     C:/UTILS/README.EXE
17720       m..   311934     C:/WINDOWS/MOUSEVKD.386
725504      m..   311940     C:/WINDOWS/ERGODEMO.DLL
11872       m..   311933     C:/WINDOWS/MOUSE.DRV
…
```

Wed Nov 03 1993: Microsoft Works Installed

```
149         m..   1653262    C:/MSWORKS/SETUP/SETUP.INI
13949       m..   1653259    C:/MSWORKS/SETUP/_MSSETUP.EXE
19023       m..   1544716    C:/MSWORKS/WKSTMPL/TMPL803.KS
21648       m..   425663     C:/WINDOWS/SYSTEM/CTL3DV2.DLL
22528       m..   1544733    C:/MSWORKS/WKSTMPL/TMPL804.DB
98640       m..   425669     C:/WINDOWS/SYSTEM/WKSOLE.DLL
…
```

Fri Dec 31 1993: Microsoft Windows 3.11 setup rerun -

```
436560      m..   311819     C:/WINDOWS/SETUP.EXE
43609       m..   311871     C:/WINDOWS/SMARTDRV.EXE
155538      m..   311827     C:/WINDOWS/MSD.EXE
1104        m..   425603     C:/WINDOWS/SYSTEM/MMTASK.TSK
55168       m..   311858     C:/WINDOWS/PIFEDIT.EXE
…
```

Tue May 31 1994: Upgrade of MS-DOS 6.2 to MS-DOS 6.22
Note this is consistent with the release in April 1994 of MS-DOS 6.22, an upgrade that added the disk compression package DriveSpace [PL2]

```
72842        m..   1038875     C:/DOS/SETUP.EXE
38138        m..   4           C:/MSDOS.SYS
400880       m..   1038935     C:/DOS/MWBACKUP.HLP
172198       m..   1038916     C:/DOS/MSAV.EXE
60646        m..   1038868     C:/DOS/README.TXT
66294        m..   6           C:/DRVSPACE.BIN
```
…

Mon Nov 21 1994: Installation of Premier Manager 3 game

```
80480        m..   2043909     C:/PM3/INSTALL.EXE
```

Wed Mar 15 1995: Reinstallation of CorelDraw 4.0
Note that this correlates with the CorelDraw log file described earlier.

```
1328         m..   425815      C:/WINDOWS/SYSTEM/TT0154M_.FOT
1332         m..   425814      C:/WINDOWS/SYSTEM/TT0156M_.FOT
3787         m..   2540049     C:/COREL40/CONFIG/CORELFLT.INI
3778         m..   2540047     C:/COREL40/CONFIG/CORELDRW.REG
927          m..   2540046     C:/COREL40/CONFIG/CORELCHT.REG
3118         m..   2540038     C:/COREL40/CONFIG/CORELPNT.REG
```

Many file deletions are noted in the period of 1995-1996 and indicate deletion mainly of Word Documents for example,
Sat Dec 21 1996:

```
2618  m..   91724 C:/WINWORD/_ET#CHAR.DOC (deleted)
2618  m..   91724 <cav.hdd1.110803.dd-_ET#CHAR.DOC-dead-91724>
```

The timeline showed absolutely no activity at all for 1997 – this indicates that the system may have been replaced by a newer model. The absolute last activity is after a gap of some 18 months:
Sat Jun 27 1998: Access of tactical data for Premier Manager 3 soccer game

```
1952  m..   2043930     C:/PM3/TACTDATA.DAT
```

This shows that the user had probably loaded the Premier Manager onto the new PC system but wanted to restore saved data from the old game play.

Recover deleted files

Many files that had been previously deleted and thought to be gone forever were viewable and recoverable using Autopsy. The "view all deleted files" option under the file analysis menu showed around 200 files, a sample of which is shown in Fig. 5. The majority of the files were Microsoft Word (`.doc`) documents containing sensitive personal and business information.

A cross section of example documents are given in the following sanitised extracts:

<u>Deleted Extract 1: Bank details</u>

File _CCOUNTS.DOC deleted on 17/12/1996, inode 48

Further to my telephone conversation with you I confirm that our account with the _______ Bank has now been moved to _________, please can you transfer all future payments to this branch until further notice. The details are as follows:-

Bank Address, Sort Code - xx-xx-xx, Account no – xxxxxxxx, Account Name – Mr & Mrs J. X

r / r	C:/WINDOWS/TEMP/_SPL283D.TMP	1995.03.17 11:45:32 (GMT)	1970.01.01 00:00:00 (GMT)	1970.01.01 00:00:00 (GMT)
r / r	C:/WINDOWS/TEMP/_SPL2815.TMP	1995.03.17 11:45:32 (GMT)	1970.01.01 00:00:00 (GMT)	1970.01.01 00:00:00 (GMT)
r / r	C:/WINDOWS/TEMP/_SPL2A27.TMP	1995.03.17 11:45:32 (GMT)	1970.01.01 00:00:00 (GMT)	1970.01.01 00:00:00 (GMT)
r / r	C:/WINDOWS/TEMP/_SPL2A1B.TMP	1995.03.17 11:45:32 (GMT)	1970.01.01 00:00:00 (GMT)	1970.01.01 00:00:00 (GMT)
r / r	C:/WINDOWS/TEMP/_SPL2A4F.TMP	1995.03.17 11:45:34 (GMT)	1970.01.01 00:00:00 (GMT)	1970.01.01 00:00:00 (GMT)
r / r	C:/WINDOWS/TEMP/_SPL2916.TMP	1995.03.17 11:45:34 (GMT)	1970.01.01 00:00:00 (GMT)	1970.01.01 00:00:00 (GMT)
r / r	C:/WINDOWS/TEMP/_SPL282E.TMP	1995.03.17 11:45:34 (GMT)	1970.01.01 00:00:00 (GMT)	1970.01.01 00:00:00 (GMT)
r / r	C:/WINDOWS/TEMP/_SPL2817.TMP	1995.03.17 11:45:34 (GMT)	1970.01.01 00:00:00 (GMT)	1970.01.01 00:00:00 (GMT)
r / r	C:/WINDOWS/TEMP/_SPL2F3E.TMP	1995.03.17 11:45:36 (GMT)	1970.01.01 00:00:00 (GMT)	1970.01.01 00:00:00 (GMT)
r / r	C:/WINDOWS/TEMP/_SPL2F13.TMP	1995.03.17 11:45:36 (GMT)	1970.01.01 00:00:00 (GMT)	1970.01.01 00:00:00 (GMT)
r / r	C:/WINDOWS/TEMP/_SPL2E2B.TMP	1995.03.17	1970.01.01	1970.01.01

ASCII (display - report) * Strings (display - report) * Export * Add Note
File Type: data

Figure 5: Output from view all deleted files in case #2, using the file analysis feature of the Autopsy forensic browser

Deleted Extract 2: Emotional letter
File _ETTER.DOC deleted on 4/12/1996, inode 49

I know that you feel I have ruined your life to live my new one. That I left my children…
…we had a lot of stress due to amount of effort we put into building our life style we started with practically nothing and achieve a lot materially…I have had to learn the hard way. We all make mistakes.…awful atmosphere at home …If he loved me like he said, he could not fall in love with anyone so easily and quickly. and sacrifice all those years on a new relationship….you have told me to leave you a lone and not contact you - I find that hard to do but if that is what you truly want I have to respect your wishes. I will not phone or contact you again unless you contact me first. I will always be there for you, I love and miss you very much.

Deleted Extract 3: Company finance procedure
File _AYE.DOC deleted on 17/12/1996, inode 39

Each month I would enter all cheques regarding expenditure for the Ltd Company. ….I then worked out the vat on each item… I entered all banking details, amounts received, interest, fees charged by bank, etc.

Deleted Extract 4: Pension data
File _BLETTER.DOC deleted on 13/9/1996, inode 47

To: XXX Bank
Re: Pension Account Nos - xxxxxxxx & xxxxxxxx

Please note that all future correspondence should be send to the home address as above.
Please can you clarify the withdrawal from account no xxxxxxxx on the 11 July 1996
Please contact me on xxxxx xxxxxxx to clarify the position.

Yours sincerely,
Mrs X

These examples are a small representative sample. Many other sensitive, deleted documents could be recovered, including CV/resumes, business, school projects and personal family documents.

Strings search
Using the `strings` command in combination with the `fgrep` command allows us to delve into a large image file directly and match for multiple interesting keywords in a patternfile. Usage:

```
#  strings  imagefile | fgrep keywords –f patternfile
```

where the pattern file contains one string per line e.g:

```
virus
trojan
rootkit
etc…
```

Depending on the nature of the forensic investigation, different sets of strings in the patternfile would be used. For example, in a child pornography investigation one would look likely for strings within the disk image such as "`lolita,teens,kiddie,kiddie porn, underage,…`".
In the case of a suspected hacking compromise strings such as "`hacker,infected,rootkit,virus,trojan,backdoor,…`" would be useful.

An alternative method used here is to use the powerful string search functions of Autopsy.
In the present case, searching for strings containing the keyword '`hacker`' only produced reference to documents containing '`Whackers`': building industry tools used in property development! No results were returned for any of the search strings except '`infected`' and '`virus`' that referenced the same document:

```
boot sector viruses removed :
Total Files checked :
Total File viruses found :
Total File viruses removed :
END OF REPORT.         $
Virus %s was found in file:%s
$File was cleaned - virus destroyed
%s Boot sector virus was found in drive %c:.
```

```
Boot sector was cleaned - virus destroyed.
```

This looks like a template output for the integral MS-DOS anti-virus scanning program. The text `%s` is often used in programming as a generic place holder for ASCII text strings.

The system under study was not connected to the Internet and was used as an office business machine. Communication with clients and customers were conducted using fax and conventional mailing. At the time of the system's active life, 1992-1996, most home PC's in the UK did not have the luxury of Internet connectivity which was pretty much limited to large corporates and universities. As such direct hacking incidents were not a concern to small business and home main PC user and is reflected in the absence of strings matched.

As we have seen, operating systems, upgrades and programs were installed via a number of floppy disks used in succession. Therefore, the most efficient way of spreading viruses was via distribution on infected floppy disks that, in turn, often infected the hard disk boot sectors[12].

Due to the dominant business usage of the system, a string search for '`fraud`' seemed appropriate and referenced a document apparently referencing a solicitor who had been '`imprisoned for fraud, the rest of the partners formed company Y`'. Although this information is incidental in this investigation, it proves that a wealth of information that can be extracted from a simple one word string search.

Conclusions

The forensic case study detailed in this report was chosen as representative of the type of computer system that can be found in the public domain, anywhere in the world. The computer system was bought from a public refuse dump in the UK and the hard drive was found later to be in perfect working order.

Despite the fact that the owners had deleted document files and condemned the system as trash, once subjected to computer forensic examination, a plethora of demographic, personal, social and business information were recovered from the hard drive. This is not an isolated, unique case.

During this study, the history of the system could be clearly reconstructed. The system was clearly used for predominantly business. School dissertations and CV's were recovered that implied that the daughters had access and that it would most likely have been sited at the family home. The system was technically unsophisticated by today's standards, and was running the old Windows 3.1 operating system. Typical of small business computers of this time period, there was no Internet, network connection or email usage.

[12] The web-archived virus bulletins make interesting reading and details in-vogue viruses e.g., Michelangelo [VB]. The virus timeline page make interesting reading in the social history of computer viruses in the era before macroviruses dominated the news [VIR].

Additionally, the analysis provides us with an insight into the lives of the owners and their social and economic details.

It was also apparent that the couple were married, most likely in their early fifties with two daughters, and that they owned a property development business. In the building of the business, family life had been sacrificed and one daughter had left the family home and had fallen out with the mother. The father had an affair and the parents subsequently were going through a divorce where financial arguments and difficulties ensued.

Although the data is historic, it may still be gold dust to hackers.

It is important to realise that we must understand the methods hackers use in order that we can secure our own systems and data. To quote ancient Chinese military strategist, Sun Tzu in Chapter 3 of the famous Art of War: "*Know your enemy and know yourself; in a hundred battles, you will never be defeated. When you are ignorant of the enemy but know yourself, your chances of winning or losing are equal. If ignorant both of your enemy and of yourself, you are sure to be defeated in every battle*." [TZU].

This type of social and financial data could be used to aid identity theft, social engineering attacks or to hold the 'innocent' to ransom via blackmail. Social engineering techniques are described in detail by Mitnick [KM]. Basically, the social engineer uses deception and persuasion techniques to gain further information as a platform for further criminal activities. Such information may include financial transactions, account passwords, or other sensitive information.

As can be clearly seen, this type of information can be obtained relatively easily in public domain systems. All this data was was obtained from one very small, old hard drive left by the owners as trash at a public refuse dump for anyone to legally buy.

Guidelines for secure disposal of computer systems and data

Following this study, methods are detailed that can be used help secure computer hard drives and systems that are no longer needed.

As proven, files that are deleted from public domain systems are still easily recoverable using standard computer forensic methods. One important conclusion for securing data in our systems is the need for a good **corporate computer system and information disposal policy**. While most companies have already implemented acceptable use, computer usage and security policies, very few have an active policy for the secure deletion and trashing of systems that are no longer needed or to be disposed. I propose that a strong policy should include the following guidelines:

Policy scope:

The company may possess computer equipment that is no longer required due to excess wear, obsolescence, damage, new installations or excessive cost of maintenance

Procedural guidelines:

- **Establish standard methods of electronically wiping data to be disposed**
  Many tools exist to perform secure file deletion of hard drives depending on the level of wiping required. Using the Linux `dd` command with the zero filler buffer `/dev/zero` is a free way of overwriting a drive with zero's. More complex commercial products often use repeatedly overwriting the drive with random data, often to military and governmental standard. Examples of wiping utilities are BCWipe [BCW], Eraser [ER], Paragon Disk Wiper [PDW] and DBAN [DB].

- **Donation of systems to schools, charities or to any other public body must have data backed up and be securely wiped of any corporate data**

- **Any corporate paperwork and notes should be shredded efficiently in a cross cutting "confetti" paper shredder wherever possible**

- **No public access to company trash or dumpsters should be allowed and dumpsters should be secured within the corporate premises and locked outside of office hours**
  Hackers, perfectly legally, search unsecured dumpsters placed outside on public streets to find sensitive information such as transactions, personal information, computer passwords and network details. Placing them within an area that the company legally owns will at least enforce a trespass on such attempts.

- **Efforts should be made to reuse equipment if useful life has not been exceeded**

- **Computer systems/hard drives that have exceeded their useful life are to be backed-up and physically destroyed or given to an appropriate company specialising in data disposal**
  Responsible businesses will use professional trusted computer disposal companies to destroy information systems permanently.

  A non-commercial and unsophisticated solution to drive destruction is the hammering the drive into tiny fragments with a sledgehammer ensuring that it cannot be read.
  As an aside, it should be noted that, in environmental terms, computers and other hi-tech equipment contain various environmental hazards including cadmium. In Europe alone, six million tons of electronic waste are clogging refuse dumps and the European Union (EU) is proposing that electronics producer's take back their waste products [WAS]. It is worth checking local laws, as it may not even be legal in your state/county/country to dump computer systems and peripherals.

5. Part 3 - Legal Issues of Incident Handling

The following issues relate to United Kingdom (UK) law and are current as of November 2003.

UK Law: an extremely high level overview

The UK comprises Great Britain and Northern Ireland and is a member of the EU. As such, increasing presence of EU law into UK law is being noted. The primary statutory legislation for criminal proceedings is the UK 'Acts of Parliament' with secondary legislation enacted in the form of orders and regulations. Law enforcement will generally get involved with criminal cases and often involves imprisonment. Specific statutes can be applied or removed from the main Acts within Wales, Scotland and Northern Ireland. Civil law generally applies to monetary compensation for damage or loss.

Based upon the type of material John Price was distributing, what if any, laws have been broken based upon the distribution?

John Price in case#1 was found guilty of distributing copyrighted material via the corporate network. These files were shown to be MP3 audio files distributed on publicly available networks for file sharing and potentially on sold on recordable CD's (CD-R's) holding some 150 audio files each. As well as violating the acceptable use policies of the Company and its Internet Service Provider (ISP), the distribution and sale breaks several primary UK laws on several accounts.

Music piracy encompasses any illegal recording of copyrighted music works and is an extremely serious problem in the UK. The British Phonographic Industry (BPI) estimated that the total estimated size of piracy market in 2000 was equivalent to a retail value of £20,500,000 ~($35,000,000) [BPI].

MPEG-1 Audio Layer III (MP3) is a standard format for compression and decompression of digital audio file. Using software, it is trivial to encode or 'rip' a CD of audio files and store them at CD quality to be played back by a software decoder. These files are copiously shared in the public internet via popular peer-to-peer (P2P)networks such as Kazaa, Gnutella and Napster. The Record Industry Association of America (RIAA) has filed lawsuits against 261 P2P file swappers with copyright infringements [RIA].

While MP3 is not an illegal format, the creation and distribution of MP3 files of commercial audio most seriously contravenes the **Copyright Designs and Patents Act (CDPA) 1988**. This Act is complicated and protects a wide range of copyrighted material. The maximum penalty is two years imprisonment. Copyright is the property right affecting a wide range of original works including sound recordings, films, and musical works. Any infringement of the Act is always a criminal offence, period. Primary infringements of the CDPA occurs on two accounts in this case: firstly when the original work is copied without permission of the copyright owners (the record company, performers, artist) and secondly when the material is distributed onto the public domain [CDP]. Secondary infringements may include the involvement in provision of methods for making infringing copies (CD burning) and handling infringed copies.

The compilation onto recordable CD's and subsequent potential sale constitutes **piracy**: the unauthorised duplication of an original recording for commercial gain. Since these fake CD's represent false representatives of the original product and are inferior in quality than the consumer would expect, the production may contravene the **Trade Marks Act 1994** and the **Trade Descriptions Act 1968** [TMA,TDA]. If found guilty of violating the Copyright, Designs and Patents Act and Trade Descriptions Act John Price could be subject to a maximum sentence of 2 years imprisonment. Additionally, standard procedure's are to seize and forfeit infringing material and also is to seizure and forfeiting of business or home computer's involved. If he is found guilty on indictment under the Trade Marks Act could lead to 10 years in prison.

What would the appropriate steps be to take if you discovered this information on your systems? Cite specific statutes.

Various important UK laws relate to computer incidents at work, in particular those that relate to monitoring have to be complied with. The **Human Rights Act (HRA) 1998** incorporates the **European Convention on Human Rights (ECHR, 1950)** of which article five states "Everyone has the right to liberty and security of person" and article eight states an individual's right of "respect for his private and family life, his home and his correspondence". These equate to the US Constitution Amendment IV [ECHR]. The immediate relevance of this act to this case is that if monitoring of network or internet/email usage is taking place then employees must be clearly advised. This should be placed in the corporate acceptable use policy. The policy should strictly demarcate the boundaries of privacy/personal. Privacy is linked to the **Data Protection Act (DPA) 1998** that provides protection in relation to the processing, collection and retention of personal data [DPA].

In terms of monitoring communications, the most important piece of legislation is the **Regulation of Investigatory Powers (RIPA) Act 2000.** RIPA facilitates cybercrimes greatly and generally, it establishes wide rights to monitor communications data. Part I deals with interception of communications data and applies to IP addresses, email addresses, phone numbers, logs, website URL's [RIP1].

These rights include monitoring for the purposes of investigating the unauthorised use of the Company's telecommunication systems, for existence of facts or for the purposes of detecting crime and protecting national security. So monitoring is lawful on the conditions above provided that "all reasonable" efforts have been made to inform every person using the system that they are being monitored. Additionally, in the interests of preventing crime, the Company can provide evidence to law enforcement. This is included in the Company policies and on warning banners placed on computer systems and brought to staff attention.

Clearly monitoring and computer investigations can proceed without the need for a warrant and permission to search from the employee should be asked in the presence of several witnesses. This improves admissibility of evidence and produces a psychological advantage.

Using webserver logs can identify employees web surfing habits and identify sites where steganography/data hiding tools such as bmap and identify the upload and download of mp3 files. The origin of the crime should be identified by the IP address of the suspect machine, the identify of all egress and ingress traffic from this user should recorded. Prior to examining logs, the server should be imaged and logs examined to avoid overwriting potential evidence.

If the crimes were detected "live", then logging should be ensured to be on so that an admissible evidence trail is recorded. The timezones and date/time would be identified for each log so as to be able to correctly correlate this information with system computer and remote webserver logs.

Any direct evidence such as seized hard drives, floppies, computers and printouts investigations should be gathered. Juries and judges like as much direct evidence as possible. This direct evidence and best evidence disk images and logs taken should be clearly tagged and good notes taken throughout. All evidence would be placed in a locked place away from physical or electromagnetic disturbances. A company safe with secure access would be ideal.

The criminal activities of the suspect would be reported to the appropriate contacts within the organisation including the MD and security manager. It would be important to investigate the other party, Mike, in order to prevent further criminal activities or destruction of evidence.
In the Company management decide to prosecute then law enforcement should be notified as soon as possible to make sure that company is not left criminally liable.

Since the legalities in light of the RIPA act and DPA are notoriously complex, the early involvement of corporate lawyers would be advisable if prosecutions are to be made. The legal counsel will also advise what information about the suspect can be given to law enforcement agencies.

In the event your corporate counsel decides to not pursue the matter any further at this point, what steps should you take to ensure than any evidence you collect can be admissible in proceedings in the future should the situation change?
It is crucial that any evidence is preserved in case proceedings occur later. The defence must not be allowed the opportunity to argue that the investigator had altered data. The data and evidence would again be stored in a secure safe or even offsite in a guaranteed secured safe.
Without a record of the md5 hashes, the evidential chain of custody cannot be proven and the evidence is not admissible. It is prudent to advise senior management of the existence of the evidence and keep a written log of the events from seizure to storage

including individuals who handled with evidence and when. Backup copies of the server logs should also be retained in a secure place. Copies of any supplementary evidence taken by the system administrator such as network and webserver logs should also be recorded and has values taken.

Again logging should be turned on to monitor any further activity with notes of timezone and system time settings recorded for remote computer and server synchronisation.
In addition to preserving the evidence, steps should also be in place to make sure that the same situation does not happen again. This is good time to remind staff on corporate policy and educate on the dangers of file sharing and downloading.
Someone in the company should be assigned the specific responsibility of making ensuring that to help avoid incidents repeating.

How would your actions change if your investigation disclosed that John Price was distributing child pornography?
UK law on child pornography is well defined. The **Protection of Children Act (PCA) 1978,** defines child pornography as any images of children under 16 years of age, involved in sexual activity or posed to be sexually provocative [PCA]. Regarding Internet child pornography, the Internet Watch Foundation (IWF) oversees and performs investigations in collaboration with law enforcement and ISP's [IWF].

Section 160 of the CJA makes the simple possession of child pornography a serious arrestable offence carrying a maximum prison sentence of five years.

The most serious charge under the Section 1 of the PCA was amended by the **Criminal Justice and Public Order Act 1994** Part VII Obscenity and Pornography and Videos, to include the making or allowing permission to make incident images of children or "*pseudo-photographs*": an image made by computer graphics or otherwise that appears to be a photograph and includes "*data stored on a computer disc or by other electronic means*" [CJPO]. After the landmark case *Regina vs. Bowden* (1999), the term "*make*" was defined to include child pornography images that are downloaded from the Internet and stored or printed. This carries a maximum prison sentence of ten years [BOW].

The **Obscene Publications Act 1959** leaves the court to decide what constitutes 'obscene' but would be generally include images featuring extreme sexual acts and that is likely to "*deprave and corrupt*" those likely to read, hear or see the matter contained within.

There have been a number of high profile arrests made in Britain for possession of child pornography. These include music industry figures Gary Glitter (Paul Gadd), Derek Longmuir of the Bay City Rollers and Pete Townshend of The Who.
Despite the maximum prison sentences given above, UK case law precedence has produced lighter sentences. Gadd had pleaded guilty of downloading some 4,000 images of child pornography and was given a fourth month prison sentence, of which he served only two [PG]. Longmuir, a resident of Scotland, had downloaded child

pornography and claimed he had saved it to disk apparently for an American friend. He pleaded guilty and was given 300 hours community service [DL]. Townshend was investigated during the high profile FBI Operation Avalanche against Landslide Productions, a global aggregator of child pornography websites [OA].
In a sub-investigation in the UK termed 'Operation Ore', Townshend, had been identified by credit-card information as having accessed a child pornography website. Townshend's defence that he was only performing research into child abuse for his upcoming autobiography was later accepted [OO].

It should be noted that a lot of the child pornography servers are sited in countries in the Far East where child pornography is not illegal. Nevertheless, in UK law, any downloading or uploading from remote sites of child pornography would constitute criminal offences under the CJPO Act by the making/downloading of the images. Moreover, in the UK it is a criminal offence to deliberately search for any form of child pornography regardless of the storage and distribution.
In terms of the current case, a forensic investigation must absolutely ascertain whether the suspect had definitely accessed a remote site for the acquisition of child pornography. This is particularly important in view of two recent UK cases where two UK men were separately cleared of charges accusing the storage of child pornography on their computers. They were in fact determined to have been the victims of child pornography storage caused by trojan horse infections. This has coined the term "Trojan defence" [TRO1]. Trojan Defence is gaining popularity in defending UK cybercrime cases. In the recent distributed denial of service attack (DDoS) on the Port of Houston's IT systems, a UK teenager was found not guilty on Trojan defence [TRO2]. It will be interesting to observe future case law to see whether Trojan defence is successfully and whether there will be any amendments to the UK cybercrime laws to include trojan and virus infections as precursors to crime.

On the immediate discovery of the child pornography, it is important to notify the police. Additionally, corporate lawyers should be consulted. Under the RIPA Act, the Company's ISP can initiate law enforcement involvement directly in their role as a public telecommunications provider since there is evidence of direct criminal activity.

It is unlikely that someone *distributing* something as serious as child pornography would do so without having a deeper involvement. The case would likely involve law enforcement warrants and examination of the suspect's home premises to look for further evidence. It is also unlikely that this would be a solitary affair and an investigation into the suspect's network of contacts should be made. Detailed forensic examinations of the suspect's filesystem, email and Internet usage both at home and at work would most likely have to be performed.

The investigation would potentially involve other corporate machines and external servers that may have been discovered to store or distribute illegal material. Usually, in these sort of crimes, underground paedophile web-rings exist and law enforcement would have to investigate their records and work with ISP's to identify child pornography perpetrators. Since August 2002, ISP's are obliged to comply with interception warrants under the RIPA Maintenance of Interception Capability Order.

REFERENCES

[JB] Bates, Jim. “Between Fact and Fiction”, in “Computer Forensics the emerging discipline”, British Computer Society ISSG magazine, Oxford, Spring 2003

[ACP] Association of Chief Police Officers. “Good Practice Guide For Computer based Evidence” Version 3 Online, NHTCU, 2003 http://www.nhtcu.org/ACPO%20Guide%20v3.0.pdf

[HA] Stallings, Bill. “Authentication, Hash Functions, Digital Signatures”, 22/1/98 http://islab.oregonstate.edu/documents/People/stallings/6.pdf

[TCT] Farmer, Dan and Venema, Wietse. “The Coroner’s Toolkit”, 2003 http://www.porcupine.org/forensics/tct.html

[TCT2] Carrier, Brian. “The TCTUtil’s”, 14/09/03 http://www.cerias.purdue.edu/homes/carrier/forensics/

[SK] Carrier, Brian. “The Sleuth kit download page”, 28/07/03 http://www.sleuthkit.org/sleuthkit/download.php

[LNT] Russon, Richard. “The Linux NTFS Red Hat Page”, 2002 http://linux-ntfs.sourceforge.net/info/redhat.html

[AT] Carrier, Brian. “Autopsy Forensic Browser: Description”, http://www.sleuthkit.org/autopsy/desc.php

[KER] Bovet, Daniel and Cesati, Marco. Understanding the Linux Kernel. O’Reilly, 2000

[TW] Tripwire, Inc. “Tripwire Integrity Assurance Company”, 2003 http://www.tripwire.com/

[AS] Stancin, Alexander. “Installing and Running Tripwire”, 5/10/01 http://www.vrlteam.org/home.asp?vrl=library&adv=78

[HEX] Sand, Espen. “KHexEdit - Hex editor for KDE”, 14/12/99 http://home.online.no/~espensa/khexedit/

[SNO] Caswell, Brian and Roesch, Marty. “Snort Open Source NIDS”, 3/11/03 http://www.snort.org/

[ETH] Ethereal.com. “The Ethereal Network Analyser”, 2/11/03 http://www.ethereal.com/

[LIB] polito.it. "WinPcap: Free Packet Capture Architecture for Windows",12/09/03
http://winpcap.polito.it/

[VM] VMWare Inc. "VMWare Workstation 4.0" 2003
http://www.vmware.com/landing/ws4_home.html

[MAS] Scott, Michael. "The Quest for Root – Hacker Techniques and UNIX Security", SANS GSEC Practical, 09/02
http://www.giac.org/practical/Michael_Scott_GSEC.doc

[NC] Giacobbi, Giovanni. "GNU Netcat – Official Homepage", 2003
http://netcat.sourceforge.net/

[AC] Chuvakin, Anton. "Linux Data Hiding and Recovery", 10/03/02
http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html

[ST] Westphal, Kristy. "Steganography Revealed", 09/04/03
http://www.securityfocus.com/infocus/1684

[SCY] Scyld Corp. "Index of /pub/forensic_computing/bmap", 29/01/01
http://www.scyld.com/pub/forensic_computing/bmap/

[KIT] Lenkey, Gideon. "BUILDING A JUMP KIT", 07/01/02
http://www.infotech-nj.com/papers/JumpKit_HOWTO.txt

[SYS] He, Jialong. "LINUX System Call Quick Reference", 2003
http://tiger.la.asu.edu/Quick_Ref/Linux_Syscall_quickref.pdf

[CMA] HMSO. "Computer Misuse Act 1990 (c.18)", 20/09/00
http://www.hmso.gov.uk/acts/acts1990/Ukpga_19900018_en_2.htm

[PSS] PacketStorm Security. "linux/security", 2003
http://packetstormsecurity.nl/linux/security/

[DD] AntiOnline. "What is Dumpster Diving?", 2003
http://www.antionline.com/fightback/What_Is_Dumpster_Diving.php

[DIT] Dittirich, Dave. "Basic Steps in Forensic Analysis of Unix Systems",
http://staff.washington.edu/dittrich/misc/forensics/

[PM3] Chown, Tim. "Premier Manager 3 – Review", 2003
http://www.gamesdomain.com/gdreview/zones/reviews/pc/pm3.html

[PL1] PowerLoad. "DOS Timeline Part I", 18/07/03
http://www.oldstuff.myagora.net/powerload/timeline.htm

[PL2] PowerLoad. “DOS Timeline ~ Part Two - 1994 Onwards”, 18/07/03
http://www.oldstuff.myagora.net/powerload/tmeline2.htm

[TZU] CUNY. “Chinese Cultural Studies: Sun Tzu: The Art of War”, 2003
http://academic.brooklyn.cuny.edu/core9/phalsall/texts/artofwar.html

[VB] Wilding E. Editor Virus Bulletin, March 1992
http://www.virusbtn.com/magazine/archives/pdf/1992/199203.PDF

[VIR] EXN Canada. “The History of Computer Viruses - A Timeline”, 04/05/03
http://www.exn.ca/nerds/20000504-55.cfm

[KM] Mitnick, Kevin D and Simon, William L. The Art of Deception: Controlling the Human Element of Security, Wiley, 2002

[BCW] Jetico Inc. “BCWipe”, 2003
http://www.jetico.com/bcwipe.htm

[ER] Heidi Software Ltd. “Eraser”, 2003
http://www.heidi.ie/eraser/default.php

[PDW] Paragon Software. “PARAGON Disk Wiper”, 2003
http://www.disk-wiper.com/

[DB] Source Forge. “Daril’s Boot and Nuke”, 2003
http://dban.sourceforge.net/

[WAS] De Bony, Elizabeth. “EU proposals aim to reduce electronic waste”,15/07/00
http://www.cnn.com/2000/TECH/computing/06/15/electronic.waste.idg/index.html

[BPI] BPI Estimates. “ESTIMATED SIZE OF UK PIRACY MARKET 2001”
http://www.bpi.co.uk/piracy/piracytable.html

[RIA] Borland, John. “RIAA sues 261 swappers”, 09/09/03
http://news.zdnet.co.uk/business/legal/0,39020651,39116203,00.htm

[CDP] HMSO. “Copyright, Designs and Patents Act 1988 (c. 48)”, 20/02/00
http://www.hmso.gov.uk/acts/acts1988/Ukpga_19880048_en_3.htm

[TMA] HMSO. “Trade Marks Act 1994”
http://www.hmso.gov.uk/acts/acts1994/Ukpga_19940026_en_2.htm

[TDA] DTI. “Guidance Note - The Trade Descriptions Act 1968”, 2003
http://www.dti.gov.uk/ccp/topics1/guide/tda1968.pdf

[ECHR] University of Essex. “ECHR 1950”, 20/03/2003
http://courses.essex.ac.uk/lw/lw655/ECHR%201950.htm

[DPA] HMSO. “Data Protection Act 1998 Ch. 29”, 24/07/98
http://www.hmso.gov.uk/acts/acts1998/19980029.htm

[RIP1] HMSO. “Regulation of Investigatory Powers Act 2000”
http://www.hmso.gov.uk/acts/acts2000/00023--b.htm

[IWF] IWF. “UK Law Covering Remit”, 2002
http://www.iwf.org.uk/hotline/CHILD

[CJPO] HMSO. “Criminal Justice and Public Order Act 1994”
http://www.hmso.gov.uk/acts/acts1994/Ukpga_19940033_en_8.htm

[BOW] Cyber-rights.org. “R v JONATHAN BOWDEN (1999)”, 2003
http://www.cyber-rights.org/documents/rvbowden.htm

[PG] BBC News. “UK Anger at Glitter sentence”, 13/11/99
http://news.bbc.co.uk/1/hi/uk/518175.stm

[DL] BBC News. “Former Roller sentenced for child porn”, 24/03/00
http://members.aol.com/buesken/bcr/news/bbc2/no240300.htm

[OA] BBC News. “Operation Avalanche: Tracking child porn”, 11/11/02
http://news.bbc.co.uk/1/h11i/uk/2445065.stm

[OO] Wearden, Graeme. “Police swoop on Internet paedophiles”, 20/05/02
http://news.zdnet.co.uk/internet/0,39020369,2110524,00.htm

[TRO1] Kotadia Munir. “Trojan horse found responsible for child porn”, 01/08/03
http://new.zdnet.co.uk/zdnetuk/news/internet/security/0,39020375,39115422,00.htm

[TRO2] Kotadia Munir. “Trojan defence acquits British teenager”, 17/10/03
http://news.zdnet.co.uk/business/legal/0,39020651,39117209,00.htm