

Introduction to Linux Forensics

By Chris Marko
June 2005

I.	Abstract	3
II.	Mounted File systems	4
III.	Forensics	5
IV.	Common Problems	14
V.	Piecing It All Together With Examples	16
VI.	Conclusion	20
	Bibliography	22

I. Abstract

This document serves as an introductory to the tools used when performing basic forensic analysis within the Linux environment. This document assumes you have a fundamental understanding of Linux, and will not go into great details on the basics of the operating environment. It is assumed you already have installed Linux, and understand how to launch a shell console as root. You should also have a broad familiarity with the file and folder structure of the system, and how commands are executed within a shell. Tools often used under Linux that are covered in this document will be done so with forensics analysis in mind.

Linux is UNIX-like free operating system that is supported and maintained by a number of developers across the world. A number of organizations put together their own Linux distributions, which include the free kernel along with bundled utilities and applications that run on the platform. Some of these programs may be proprietary to the specific distribution, while most are generally freely available. Each distribution makes their own decisions on how certain aspects are setup, and as such, the version of a utility, syntax of a particular command, or even location of a file might vary from one distribution of Linux to the next. Most of the commands referenced in this paper should be in the system path, so you need not be concerned with the actual location of the utility files.

For the aspect of this paper, I will mainly be concerned with performing my work under the commercial release of the Novell SUSE 9.1 Linux distribution. This is one of the more user-friendly Linux distributions out there. You can find out more about it at <http://www.novell.com/linux/suse>. These commands should just as easily function under most other mainstream Linux distributions, such as Red Hat or Mandrake.

Another useful distribution which concerns itself more specifically with forensics is the F.I.R.E. portable bootable CD-ROM based Linux distribution. It includes all of the utilities covered in this paper, plus many additional forensic specific utilities. More information is available at <http://fire.dmzs.com/>. However, since we are discussing generalized Linux utilities, we will not delve into the specifics of this forensic distribution.

If you require assistance on installing SUSE, please refer to their helpful web site and installation instructions. To those who are easily intimidated, one might find that it is actually faster, and debatably easier, to install than this newer Linux incarnation of Windows. In terms of component selection, since disk space is so cheap today, I pretty much select everything. Whether or not you elect to install everything is insignificant, as you can easily install individual applications or entire suites later.

The Linux kernel itself is regularly updated and maintained by free software developers from around the world. The source code is freely available on the Internet for anyone to download and install. You can even make up your own Linux distribution if you were so inclined! (1) As you learn by studying cryptography, the safest algorithms in the world are those that are open for review and criticism by knowledgeable peers. An operating system is no different. As soon as a bug or exploit is discovered in Linux, the source code is available to the individual. That individual has the opportunity to either fix the problem and submit this fix to the Linux kernel team, or tell the world about it so that some other kind soul can fix it.

Another great feature about this openness with Linux is that anyone can easily develop their own utilities, applications, and drivers to work in Linux. This is why you see an amazing depth of possibilities in such things as the wide support for so many file system formats, to what specific types of utilities and kernel components might be the most beneficial to you. Where Windows tries to hide many components and features in the name of usability, Linux leaves many of these options readily available just waiting for you to take full advantage of. Additionally, if you do not see a particular utility or component that you want, all the tools are freely available for you to take some time and make it yourself!

II. Mounted File systems

The mount command carries with it a slight shroud of mystery to those new to the Unix world. Mounting is the process of attaching a file system found on a device to the present directory structure and file system. Some examples of devices might be a CD-ROM drive, or a Zip drive. From a forensics perspective, this might be an image of a hard drive, which is covered later. Each mounted file system is given what is known as a mount point, or a directory in the directory tree on your file system, from which it can be used.

You can have a file system automatically mount at boot-time by adding it to the File System Table. Alternatively, you can also mount file systems by using the mount command. To see a list of the file systems that are actually mounted at anytime, simply issue the `mount` command with no switches. (2)

File systems that are described in the `/etc/fstab` file are automatically mounted when your computer starts. The `/etc/fstab` file follows a very specific structure. Here is an example `/etc/fstab` file:

```
# cat /etc/fstab
# Device to mnt      mnt point fs type          mount options          defaults          1 1
#/dev/hda3          /           /           reiserfs          defaults          1 1
/dev/hda2           /           /           reiserfs          defaults          1 1
/dev/hda1           swap        swap        swap              pri=42            0 0
devpts              /dev/pts   devpts     devpts            mode=0620,gid=5   0 0
proc                /proc      proc       proc              defaults          0 0
usbdevfs            /proc/bus/usb usbdevfs    usbdevfs         noauto            0 0
/dev/cdrom           /media/cdrom auto        auto              ro,noauto,user,exec 0 0
/dev/fd0             /media/floppy auto        auto              noauto,user,sync  0 0
```

Notice the line that starts with `#/dev/hda3`. The pound sign is a comment symbol, which tells Linux, when reading this file, to basically ignore this entire line. If I remove the pound symbol, the system will try and mount `/dev/hda3` upon boot-up. With the `#` symbol in place, it will ignore this.

The `mount` utility is used when logged in to a shell to manually mount additional devices that may not be automatically mounted during boot time. The `mount` utility can recognize more than 30 different file system types! Examples on using `mount` will follow shortly.

III. Forensics

Linux has an amazing breadth of tools available to the computer forensics investigator. It is also highly flexible in terms of user control. This can require quite a bit of time to master, as the power often comes from the command line interface, which is not as intuitive as the modern graphical user interface that most are accustomed to.

Many common tools that you would use on a Linux platform come built-in to the Linux distribution. While many new forensics-specific tools are being released, what is covered here offers what should be an excellent baseline to taking advantage of the existing basic tools easily available at your fingertips. It is beyond the scope of this paper to cover each tool in detail. As such, it is recommended that as you use each tool, you perform additional reading to fully understand how to take advantage of the tool to its fullest potential. When unsure, the man page is a great place to start. Just type `man toolname`. `man mount` is a great place to start, so you can learn about all the different file systems that the `mount` utility supports.

One thing to also remember when approaching any computer system for live analysis in its current active state is that no tools on the system itself can necessarily be trusted. To perform the requisite analysis, the seasoned computer forensics investigator may supply their own toolset on a CD. Of course, this is not quite so simple with Linux, as it comes in so many different variations. A tool compiled under Red Hat 8 may not necessarily run on Mandrake 10 due to some conflicting software versions, or even library dependencies. Each distribution uses its own versions of files, and even places them in different locations. To help minimize any software conflict issues, you should do your best to try and learn what specific Linux operating system flavor you will be approaching, and get the appropriate tools compiled for that version beforehand. If unable to supply your own tools or if they are problematic, then you may just have to do your best with the tools on the system itself. If forced to use the tools on a live system, obtain MD5 and SHA hashes of the tools you used to later determine their integrity against known good versions. Obtaining a file hash is covered later in this paper.

Before we begin, one should have an adopted methodology of organizing data and results from an analysis. Typically this may include a standardized separate folder on a disk independent of the one you are performing analysis against. For the examples used in this paper, we use the /evidence folder on our own trusted computer. To create this folder on the local computer:

```
# mkdir /evidence
```

Remember, you do **not** want to create this directory on the suspect source disk, as you will end up overwriting existing information! Any result text output files that you wish to save during the analysis phase should be directed to save in this directory for consistency purposes.

When performing an analysis, it is also advised you be logged in as root. This will ensure you have fullest of permissions and necessarily access possible during your forensic analysis.

One simple tool available when determining the structure of a disk attached to your system is the **fdisk** utility. This utility is used for creating and recovering partition tables.

The basic fdisk commands that you should be aware of are:

- p** print the partition table
- n** create a new partition
- d** delete a partition
- q** quit without saving changes
- w** write the new partition table and exit

Note that any changes you make to a partition table will not actually take affect until you issue the write (w) command. (3)

From a forensics perspective, you will probably be most concerned with the print command to view an existing partition table on a disk.

A sample partition table may appear as follows:

```
# /mnt/cdrom/fdisk /dev/hdb
Command (m for help): p
```

```
Disk /dev/hdb: 64 heads, 63 sectors, 621 cylinders
Units = cylinders of 4032 * 512 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/hdb1	*	1	184	370912+	83	Linux
/dev/hdb2		185	368	370944	83	Linux
/dev/hdb3		369	552	370944	83	Linux
/dev/hdb4		553	621	139104	82	Linux swap

The top line shows the geometry of the hard drive, as Linux sees it. As a hard drive will have a head on each side of the platter, the disk in this example, as presented to the operating system has 32 double-sided platters with one head on each side. Each platter has 621 tracks. The same track on all disks is called a cylinder. Each track is divided into 63 sectors. Each sector contains 512 bytes of data. Note that this may not be what the physical characteristics of the disk actually are. (3)

If you want to view similar information straight from the command line, you can also use the **fdisk -l** command. Here is an example:

```
# /mnt/cdrom/fdisk -l /dev/hda
Disk /dev/hda: 10.2 GB, 10239860736 bytes
255 heads, 63 sectors/track, 1244 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1            1          131     1052226   82  Linux swap
/dev/hda2            *        132        1244     8940172+   83  Linux
```

When looking at the above example, notice the last column labeled System. This column can provide an important clue as to the type of file system is occupying that part of the disk.

This example is helpful in showing you how to direct this output to a file:

```
# /mnt/cdrom/fdisk -l /dev/hda >/evidence/fdisk.disk1
```

This will output what we saw in the previous command directly to the file `fdisk.disk1` inside of our evidence folder.

Here is another example that shows some of the different file systems the `fdisk` command might recognize:

```
# /mnt/cdrom/fdisk -l /dev/hdc

Disk /dev/hdc: 255 heads, 63 sectors, 1582 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot      Start     End      Blocks   Id  System
/dev/hdc1            1       255     2048256    b   Win95 FAT32
/dev/hdc2            *       256     638     3076447+   83   Linux
/dev/hdc3            639     649       88357+   82   Linux swap
/dev/hdc4            650    1582    7494322+    f   Win95 Ext'd (LBA)
/dev/hdc5            650    1453    6458098+    b   Win95 FAT32
/dev/hdc6            1454    1582    1036161    b   Win95 FAT
```

Notice the FAT32 partition, a Linux partition, a Linux swap partition, a Windows Extended partition, another Windows FAT32 partition, as well as a basic Windows FAT partition. (4)

To create an image of a disk, a wonderful program that is included with Linux is the dd utility. dd is a real-time tool that can be used to take in-depth images of a drive. This will basically include all bytes, even those that have been flagged as bad by the operating system. Don't be surprised if dd seems to hang a bit when attempting to read such blocks.

Here is an example of using dd to create an image of a floppy disk, and output it to image.floppy1:

```
# /mnt/cdrom/dd if=/dev/fd0 of=/evidence/image.floppy1 bs=512
2880+0 records in
2880+0 records out
```

This command uses your floppy disk drive /dev/fd0 as the input file, and uses a file called image.floppy1 as the output file. The block size is set to 512 bytes. The block size is not typically needed for most hard drives, as Linux can handle the actual block size. However, it is important to be aware that it can be specified.

Once you have created a forensic image of a disk, you should also calculate its file hash and document this. Obtaining the file hash is available later in this document.

The resourceful dd utility can also be used to take an image of physical memory for later analysis. Linux physical memory is accessible via two files, the /dev/mem file, and the /proc/kcore file. The size of the dumped memory is the size of RAM. (5)

Here is an example of using dd to create an image of physical memory:

```
# /mnt/cdrom/dd if=/dev/mem of=/evidence/image.memory1
1310720+0 records in
1310720+0 records out
```

After creating any image, it is suggested that you set the appropriate permissions on it to prevent any accidents. You can use the chmod command to do this.

Here is an example to set the file to read-only for anyone accessing the file:

```
# chmod 444 /evidence/image.floppy1
```

If the numerics of chmod confuse you, you can also try following this example which sets the read only flag to true, write and execute flags to false for User Group, and Owner:

```
# chmod ugo+r-w-x /evidence/image.floppy1
```

Alternatively, you can simply use the “a” to apply read only rights to all for the evidence folder, and recursively (-R) everything within by using the following command:

```
# chmod -R a+r-w-x /evidence
```

When applicable, you should document in your Chain of Custody log when you make a forensic image of a disk, including the application used. It is recommended you also document that you explicitly made the image read-only, which can be helpful in situations such as being under cross-examination in court.

To mount a disk, you can take a look at the following example:

```
# mount -t vfat -o ro,noexec /dev/fd0 /mnt/analysis
```

The above mounts the floppy drive to /mnt/analysis. The ro tells us to mount the file system as read-only, and to not allow execution of binaries. The -t is used to specify the file system. If you omit this, mount will take a guess at its type. You can now cd /mnt/analysis and be within the newly mounted file system.

To mount an image that was previously created, let us take a look at the following example:

```
# mount -t vfat -o ro,noexec,loop /evidence/image.floppy1  
/mnt/analysis
```

Again, this tells us to mount the file system as read-only and not allow execution of binaries. Additionally, notice the loop parameter. This is a special method within Linux to mount the image file using a loopback interface. You can now cd /mnt/analysis and be within the mounted image.

The file hash is an important step to perform in any forensic analysis. This is done to verify the integrity of your data, both before and after any analysis is complete. MD5 and SHA are the most reliable algorithms to obtain a file hash with. It is virtually impossible to modify a file without somehow altering the MD5 and SHA file hash.

When you first begin analyzing a suspect image, one of the first things you should do is obtain a file hash of the image itself, and then of every single file on that system. This should be logged accordingly for later reference if necessary. This is important, as a defense lawyer in court may later question the integrity of a file you found on a system, and ask for you to prove that the same file submitted as evidence has not been tampered with since it was originally obtained.

Remember, that a raw disk does **not** need to be mounted in order to obtain the hash. All you basically need to do is be able to see its associated file within the operating system, and you can obtain a hash against it.

Here is an example of how to obtain the SHA hash of the floppy disk drive associated as /dev/fd0:

```
# /mnt/cdrom/sha1sum /dev/fd0
c3dae2b3b034a0d63a328b84c66d444103d76b2b /dev/fd0
```

Now, if lets obtain the SHA hash value of the image we made of the floppy disk drive previously:

```
# /mnt/cdrom/sha1sum /evidence/image.floppy1
c3dae2b3b034a0d63a328b84c66d444103d76b2b /dev/fd0
```

As expected, the hash values are the same. This tells us that this is a good image of the floppy. To instead obtain the MD5 hash of the floppy drive, and re-direct this output to a file, we can perform the following:

```
# /mnt/cdrom/md5sum /dev/fd0 >/evidence/md5.floppy1
```

Now, you may not necessarily want to manually run this command for every single file on an operating system. So, instead, lets look at an example of how to obtain the SHA hash of every file on the floppy disk:

```
# /cd/dev/fd0
# /mnt/cdrom/find . -type -exec /mnt/cdrom/sha1sum {} \;
>/evidence/SHA.floppy1
```

This command, broken down, basically says to run find, starting in the current directory (as specified with the "."), any regular file (-type f), execute (-exec) the command (/mnt/cdrom/sha1sum) on all files found ({}). It then redirects this output to the SHA.floppy1 file.

We are going to assume here that you have the file system you wish to analyze mounted. If not, refer to the previous section on mounting a restored image.

First, you want to change into the root of the mounted file system. For example, if you mounted the file system to /mnt/evidence, execute cd /mnt/evidence to change directory to the mount.

Next, lets perform a basic directory listing of all files, including hidden ones, in long format, which includes permissions, date, etc. We are also going to use the -R option to recursively list all the sub-directories. We can pipe this command into the less utility to make it easier to view:

```
# ls -alR | less
```

```

.:
total 610
drwx----- 22 root    root          1264 Jun 22 10:59 .
drwxr-xr-x  22 root    root           560 Jun 22 13:12 ..
-rw-----   1 root    root            0 Jun 22 10:59 .ICEauthority
-rw-----   1 root    root            99 Aug 14  2004 .Xauthority
-rw-r--r--   1 root    root            0 Jan 30  2004 .addressbook
-rw-----   1 root    root          2285 Jan 30  2004 .addressbook.lu
<snip> ... <snip>
drwxr-xr-x   3 root    root           112 Jan 28  2004 .mcp
-rw-----   1 root    root            31 Jun 22 10:09 .mcp
lines 1-24

```

This command should allow you use your arrow keys to scroll back and forth through the list of files on the system.

The ls command also includes many other parameters that can make its output even more useful for the forensics investigator. Lets see how to recursively output all the files, including permissions, date, etc. as above. Additionally, lets see the associated inode, and sort by access time:

```
# ls -laiRtu > /evidence/file.listcomplete
```

You can also retrieve a simple list of all the files on the file system by taking advantage of the find command. Here is an example, to again run from the root of the mounted file system (/mnt/evidence in our example):

```
# find . -type f -print > /evidence/file.list
```

Now, lets say you want to see all the files on the computer that have a .gif extension. We can use the grep command to search a text file, as with the following example:

```
# grep -i gif file.list
```

A smart computer user who is trying to hide something might simply try and change the file extension to throw off an investigator. However, Linux features a handy utility called file. The file utility examines the header of the specified file and does its best to identify the type of file it is, regardless of its filename extension. For example:

```
# file /evidence/file.list
file.list: ASCII text
```

Even though the file is not named with an obvious text extension (i.e. .TXT, or .DOC), it is still able to figure out that the file is just a plain old text file.

Another example that accurately identifies a Microsoft doc file:

```
# file /mnt/analysis/test.doc
/mnt/analysis/test.doc: Microsoft Office Document
```

File can often even identify the type of file system within an image file created with dd!

Now, let's take things a step further and use the file command on all of the files of a mounted file system, and output this result to a text file for later analysis:

```
# find . -type f -exec file {} \; > /evidence/filetype.list
```

Now, to examine the output file and only see what the file command returned as matching image files, let us see the following example:

```
# cat /evidence/filetype.list | grep image
```

This will return any line that has the word "image" in it. While this may not strictly include just image files, it certainly narrows down your search criteria.

You may also want to narrow your search, to only find larger files that have changed within the past 7 days. Here is an example of using find to do so(6):

```
# find . -xdev -mtime -7 -size +20000 -print | less
```

To view files, you have the cat, more, and less utilities available at your disposal in a standard Linux system. The cat utility simply raw displays the file to the terminal. The more utility is useful for paging through a specified file one screen at a time. The less utility also works similar to more, except that it adds the functionality of backwards and forward scrolling of the specified file.

Note that when using any of these utilities against a binary file, unpredictable text may populate the screen. Some files, such as Microsoft Word files, contain a mix of text and binary data. A utility called strings can be used to extract the raw text only found in a file. Here is an example of coming the strings utility with the less command mentioned previously, to extract the text from a Microsoft Word DOC file:

```
# strings /mnt/evidence/dealers.doc | less
bjbjUqUq
Marijuana Pot Test Junkie
Hello
Marijuana Pot Test Junkie
Normal.dot
Microsoft Word 9.0
`Bew
```

This will output any raw text embedded within the dealers.doc file on our mounted evidence file system. In the case of the above example file dealers.doc, I created this in Microsoft Word and simply typed in "Marijuana Pot Test Junkie

Hello” and saved. The strings command, as you can see, what able to extract the raw text embedded within the binary data from the file.

Now, the image of the disk also includes a byte for byte copy of all the data, including slack space and unallocated space. You may want to search this space as well for information. We can see the grep command used in the following example to search an image file:

```
# grep -aib marijuana /evidence/image.floppy1
>/evidence/hits.marijuana
```

The `-a` parameter tells `grep` to process the file as if it is a text file. The `i` parameter tells `grep` to ignore case, and the `b` parameter tells `grep` to return the byte offset of a returned match.

A more useful way to perform this search might be to search for a list of words. Using a text editor either within the GUI or `emacs` or `vi`, create a raw text list of words, leaving no blank line at the end. For example, use `vi` to create a file in `/evidence/wordlist.txt`:

```
Marijuana
Dealer
Cocaine
```

Now, to use `grep` to search for any of these word hits against the image:

```
# grep -aibf /evidence/wordlist.txt /evidence/image.floppy1
>/evidence/hits.all
```

The new `-f` parameter tells `grep` to use to pull its search list from the specified file following the parameter.

Now, to see your results, lets cat the file:

```
# cat hits.all
71234:will deliver the marijuana.
```

The number at the beginning is the byte offset within the image that the match was found at. Now, we can use the `xxd` utility to see that offset:

```
# /mnt/cdrom/xxd -s 71234 /evidence/image.floppy1 | less
000476f: 0000 0080 0100 0000 0000 0000 0000 0000 .....
000477f: 0000 0080 0100 0000 0000 00d1 0100 0030 .....0
000478f: 0000 0001 0200 0000 0000 0080 0100 0000 .....
000479f: 0000 003f 0500 0000 0000 005a 0100 0000 ...?......Z....
00047af: 0000 003f 0500 0000 0000 0080 0100 0000 ...?......
00047bf: 0000 005a 0100 0000 0000 00bc 0000 0012 ...Z.....
00047cf: 0000 00ce 0000 000e 0000 00a8 0000 0000 .....
00047df: 0000 00a8 0000 0000 0000 0000 00a8 0000 .....
00047ef: 0000 00a8 0000 0000 0000 0002 00d9 0000 .....
00047ff: 004d 6172 696a 7561 6e61 2050 6f74 2054 .Marijuana Pot T
000480f: 6573 7420 4a75 6e6b 6965 0d0d 4865 6c6c est Junkie..Hell
```


bytes of the output will still be synchronized and align with the actual data that was the source.

As technology demands increase, network storage requirements are constantly growing. To keep up with this demand, many network administrators are turning to something known as a storage area network, or SAN, to store files on. While this is a massive subject that is beyond the scope of this paper, one need not necessarily be intimidated when attempting to obtain an image from a Linux machine attached to a SAN. A SAN is basically a specially designed network designed to attach computer storage devices, such as large numbers of disks, to servers. While the underlying technologies connecting the machine to the SAN are unique, at the end of the day they are still presenting the file system just as any other technology, including IDE or SCSI, would. As such, most of the disk imaging techniques discussed previously can very well be used in this type of environment for basic forensic analysis as well. The biggest limitation you will probably encounter is that the partitions on a SAN are probably significantly sized. Under rare circumstances (which will become more the norm as time goes on), partitions can be up to 6TB in size! Obviously, no one does not carry a 6TB USB disk around. Not yet.

When encountering such a situation, one should use a combination of their best decision making skills and the previous information covered on how to split an image into multiple files. 1TB USB disks now exist. While not practical in most instances, chaining 6 of these to a machine may be your only option. Or, you may wish to investigate using a utility called netcat to send the data over a network to another large storage repository.

Additionally, it will probably not be feasible to boot from your own CD to obtain an image of a SAN-attached disk. SAN's typically have special software installed on each attached computer to allow it to talk to the SAN. You will either not have these utilities on the CD, or they will not be configured correctly. As such, your only option may be to image the SAN disk space while the suspect machine is running.

Finally, something to be aware of is that many computers connected to a SAN have redundant connections. This is for both load balancing and fault tolerance purposes. So, do not be surprised if you see two device files that appear to point to the same data. That is because, well, they do!

A common practice that anyone involved in computer forensics should be familiar with is to suitably sterilize, or wipe, a standalone disk prior to receiving a suspect image or suspect data. This will ensure that no data accidentally becomes mixed together. To ensure the disk is fully sterilized, you want to write zero's across the disk, as demonstrated in the following example:

```
# /mnt/cdrom/dd if=/dev/zero of=/dev/hdb bs=4096
```

This writes zero's across the drive, from beginning to end, in 4096 byte chunks.

Now, after applying the above sterilization technique, you may want to go back and confirm that it was successful. To do so, one should take advantage of the xxd utility with the -a parameter, which tells it to auto-skip. Here is an example:

```
# /mnt/cdrom/xxd /dev/hdb
0000000: 0000 0000 0000 0000 0000 0000 0000 0000 .....
*
0167ff0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
```

As you can see from the output above, the disk is zeroed out.

V. Piecing It All Together With Examples

So, lets piece everything together with a few example situations. We will cover analysis of a floppy disk, as well as analysis of a received IDE hard disk drive.

Floppy Analysis

We can start simple, with a floppy disk. You are unsure of the origin system of the floppy, but it was handed to you for analysis related to a drug case. Luckily, you have a trusted Linux desktop in your office/lab/home, so you already have an excellent foundation to begin the analysis.

First, lets make sure the floppy disk is physically write protected to ensure that it cannot be written to, and thus altered. If you end up switching it to a write protected state, make sure you document this.

First, lets create an evidence folder to serve as a central repository for all of our collected evidence and analysis data:

```
# mkdir /evidence
```

Next, insert the floppy disk into your Linux system. Lets obtain an SHA checksum on the disk and document this into a file.

```
# sha1sum /dev/fd0 > /evidence/floppy1.sha1sum
# cat /evidence/floppy1.sha1sum
c3dae2b3b034a0d63a328b84c66d444103d76b2b /dev/fd0
```

Now, lets make a forensic image of the floppy to our hard drive. This will speed up analysis, as well as help demonstrate in court that the original floppy has not been tampered with:

```
# dd if=/dev/fd0 of=/evidence/floppy1.img bs=512
2880+0 records in
2880+0 records out
```

Now, lets obtain the SHA value of the image file, and confirm that it matches the floppy disk. This will tell us that the image is identical to the data on the floppy:

```
# shasum /evidence/floppy1.img >/evidence/floppy1img.shasum
# cat /evidence/floppy1img.shasum
c3dae2b3b034a0d63a328b84c66d444103d76b2b /dev/fd0
```

We can see that the returned value matches the returned SHA hash from the floppy itself a few lines above. This is good. Now, lets use the file utility to see if we can identify the file system from the floppy:

```
# file /evidence/image.floppy1
image.floppy1: x86 boot sector, code offset 0x3c, OEM-ID
"MSDOS5.0", root entries 224, sectors 2880 (volumes <=32 MB) ,
sectors/FAT 9, serial number 0xc45e4201, unlabeled, FAT (12 bit)
```

Since we see this is a FAT12 partition, which Linux natively supports, we can now mount the floppy disk image. To be safe, lets mount it read-only, and not allow execution of any files within the image. Since this is not a physical device, we are going to use a Linux “loopback” device which tricks the system into thinking this is a physical device.

```
# mkdir /mnt/analysis
# mount -t vfat -o ro,noexec,loop /evidence/floppy1.img
/mnt/analysis
```

Now, lets obtain the SHA hash of every file on the floppy. A defense lawyer may later question whether the data has been altered since it was received, and you want to demonstrate that without a doubt it has not.

```
# find . -type -exec shasum {} \;
>/evidence/floppy1img.shalfilehash
```

The SHA value for every file on the disk should be dumped to this text file. Now, lets take a look at a friendly list of all the files on the disk itself.

```
# cd /mnt/analysis
# ls -alR | less
.:
total 5
drwx----- 22 root    root        1264 Jun 22 10:59 .
drwxr-xr-x  22 root    root         560 Jun 22 13:12 ..
-rw-----  1 root    root          0 Jun 22 10:59 .ICEauthority
-rw-----  1 root    root       40234 Aug 14  2004 crackdealers.d
-rw-r--r--  1 root    root       543123 Jan 30  2004 addressbook
```

This will present us with a nice list of files. If it fills the screen, we can use the arrow keys on the keyboard to scroll back and forth through the list to see if anything jumps out at us as obviously appealing. Above, we see an interesting file called crackdealers.d. However, when you simply try and cat crackdealers.d,

you receive a bunch of gibberish in return. Lets see if the Linux file utility can identify the file header.

```
# file crackdealers.d
crackdealers.d: Microsoft Office Document
```

This is interesting. Since we don't have Microsoft Word installed on this machine, let us just see if we can just see the ASCII text within the file.

```
# strings crackdealers.d | less
bjbjUqUq
Great Crack Dealer
Ted 617-515-5555
This guy will trade crack for lawn furniture
Billy 603-515-5555
Normal.dot
Microsoft Word 9.0
`Bew
State of NH Department of Revenue
Concord, NH
```

Now this is interesting. Apparently, we have some information related to other drug dealers. Additionally, for some reason, information about the State of NH Dept. of Revenue is coming up in the file. Was it created and saved on one of their computers? Lets see if we can search out any further information on the floppy related to any of this.

```
# vi /evidence/searchcriteria.txt
```

Within this file, lets add the following to search for:

```
Crack
617-515-5555
Ted
603-515-5555
NH
Revenue
```

And save it. Now, lets search the image for any occurrences of those words you typed in the /evidence/searchcriteria.txt file.

```
# grep -aibf /evidence/searchcriteria.txt /evidence/floppy1.img
>/evidence/searchcriteria.results
# cat /evidence/searchcriteria.results
71234: Great Crack Dealer
94322: Boston crackdealr
```

So, it appears that another crack hit comes up from somewhere else on the image. Lets browse out to that area of the image and see what we can find.

```
# xxd -s 94322 /evidence/floppy1.img | less
000476f: 0000 0080 0100 0000 0000 0000 0000 0000 .....
```

```

000477f: 0000 0080 0100 0000 0000 00d1 0100 0030 .....0
000478f: 0000 0001 0200 0000 0000 0080 0100 0000 .....
000479f: 0000 003f 0500 0000 0000 005a 0100 0000 ...?.Z....
00047af: 0000 003f 0500 0000 0000 0080 0100 0000 ...?.Z....
00047bf: 0000 005a 0100 0000 0000 00bc 0000 0012 ...Z.....
00047cf: 0000 00ce 0000 000e 0000 00a8 0000 0000 .....
00047df: 0000 00a8 0000 0000 0000 00a8 0000 0000 .....
00047ef: 0000 00a8 0000 0000 0000 0002 00d9 0000 .....
00047ff: 004d 6172 696a 7561 6e61 2050 6f74 2054 .Boston Crackdea
000480f: 6573 7420 4a75 6e6b 6965 0d0d 4865 6c6c lr addressbook..
000481f: 6f0d 0d00 0000 0000 0000 0000 0000 0000 o...11 Clarendon
000482f: 0000 0000 0000 0000 0000 0000 0000 0000 .Back Bay..Apt6.

```

Now, that is also interesting information worth noting, as it appears that we now have an address for the Boston crack dealer.

The above examples can be used to further explore the rest of the files on this disk. You may consider copying the crackdealers.d file to a machine with Microsoft Word to read its interpreted contents.

Hard Disk Analysis

You received an IDE hard drive from a corporate client. They indicated that they believe one of their former employees was stealing company information for their own personal gain. The employee quit a little while ago to go work for a competitor. Since they quit, nobody at the company has touched the employee's former computer. You dropped by their office, found the untouched computer sitting in a corner powered off, and extracted the hard drive to bring back to your own lab for analysis. You carefully documented this in your Chain of Custody Log, in case this information ends up going to court.

Much of the same analysis we performed in the previous example can be similarly implemented with hard disk analysis.

First, as with the floppy, lets connect the hard drive to the system and make an image of it. I would like to note that hardware devices do exist out there to help make this image for you, as well as special cables that ensure the drive is only accessible as read-only. However, since this paper is geared specifically towards forensics with Linux tools, and the dd utility has been proven to be quite reliable, let us proceed without hesitation.

```
# fdisk -l /dev/hdc
```

```
Disk /dev/hdc: 255 heads, 63 sectors, 1582 cylinders
Units = cylinders of 16065 * 512 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/hdc1		1	255	2048256	b	Win95 FAT32
/dev/hdc2	*	256	638	3076447+	83	Linux
/dev/hdc3		639	649	88357+	82	Linux swap
/dev/hdc4		650	1582	7494322+	f	Win95 Ext'd (LBA)
/dev/hdc5		650	1453	6458098+	b	Win95 FAT32
/dev/hdc6		1454	1582	1036161	b	Win95 FAT

Interesting. The user appears to have been using a system that had both Windows and Linux partitions on it. It was either a dual-booted system, or maybe had both operating systems installed at different points during its lifetime.

Next, lets create an image of a partition from the disk:

```
# dd if=/dev/hdc3 bs=1k conv=sync,noerror of=/evidence/hdc3.img  
bs=1k
```

Note that in this example, we are using a 1k block size. This will vary with different disks, though 1k is a safe bet (though 1k is slower to perform an image operation if a larger block size is supported, such as 4k, or even 8k).

After this has been completed, we can obtain the SHA hash of the image to ensure it matches the original physical disk.

```
# shasum /evidence/hdc3.img >/evidence/hdc3img.shasum
```

Next, we can use the file utility to see if we can obtain further information about the file system:

```
# file /evidence/hdc3.img  
hdc3.img: Linux/i386 swap file (new style) 1 (4K pages) size  
263055 pages
```

We can see that the partition imaged was in fact a Linux swap partition. We can use the file utility against images of the other file systems on this disk as well, and it will accurately identify the type of system.

In the above specific example, you cannot actually mount a Linux swap space partition. However, for other file system types, such as what we did with the floppy disk, you could. In fact, as with the floppy, your analysis may follow a similar starting pattern. However, considering the size that hard disks can potentially be these days, you will most likely find much more information that will require time and patience to sort through.

VI. Conclusion

You should now have an introductory familiarity with performing basic forensic analysis under the Linux operating environment. It should be reiterated that these are just the starting fundamental steps, as each utility has a number of other useful features that have white papers and, some cases, entire books dedicated to them. As with all other aspects of forensics, the best way to become successful is through extensive experience. The nice thing about computer forensics is that many opportunities abound to easily work on building up these skills. As these utilities are freely available, there is no financial barrier to practicing. Borrow an old floppy disk from a friend, and practice searching it. Maybe pull out a hard

drive from an old computer, or buy a cheap used one at Ebay, and see what types of information you may be able to find. Through practice, typing these commands and running all these various utilities will soon become second nature. Your confidence and credibility will grow as you start to become familiar with the uniqueness that each data analysis presents.

Bibliography

1. *Building Linux from Scratch*, <http://www.linuxfromscratch.org/>
2. Byfield, B. *Mounting harddrives, floppydisks, and more*, <http://www.linuxvoodoo.com/resources/howtos/mounting/>
3. Lissot, A., Koehntopp, K. *Linux Partition HOWTO*, <http://www.lissot.net/partition/>
4. <ftp://ftp.hq.nasa.gov/pub/ig/ccd/linuxintro/linuxintro-LEFE-2.0.5.pdf>
5. Burdach, M., *Digital forensics of the physical memory*, http://forensic.seccure.net/pdf/mburdach_digital_forensics_of_physical_memory.pdf
6. Nemeth, E., Snyder, G., Seebass, S., Hein, T. *UNIX System Administration Handbook*, 2nd ed, Prentice Hall PTR, Upper Saddle River, New Jersey, 1995.