

Beginners Guide to Linux Forensics

By Chris Marko
June 2005

I.	Introduction	3
II.	Starting Linux	4
III.	Examining Data	4
	a. Floppy Disk Analysis	4
	b. Hard Disk Analysis	10
IV.	Conclusion	13
	Bibliography	14

I. Introduction

This paper serves as a beginners guide to introducing the reader to a particular aspect of the Computer Forensics field, which is that of performing forensic analysis from within the Linux operating system.

This document assumes you have a very fundamental understanding of Linux, and will not go into great details on how to install it, nor navigate around it. There are plenty of other great resources, both online and down at your local bookstore, for that.

Most readers will probably be much more familiar with either the Macintosh operating system, or Microsoft Windows operating system. Both of these are primarily GUI-based environments. The core foundation of Linux is primarily text based. Several GUI interfaces can sit on top of this text interface, including Gnome, and KDE. From a forensic perspective, you are going to be able to really dig in much deeper by using the old style command line interface. While the different commands and switches may sound confusing at first, once you use them a few times they really are not that difficult.

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.

I would also like to make a passing comment that there are several Linux distributions that are geared specifically towards computer forensics. One example 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/>. For those individuals who may have a Windows machine at home, but still want to try and play with some of these commands, then the above distribution could be burned to a CD, and booted from.

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 one can learn when 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. Starting Linux

When you start up your computer with Linux, our goal here is to get to what is known as the console prompt. This is basically known as a shell within the Linux world. Such shells are called Bourne, Bash, and the C-shell. Each has a slightly varying look, and special parameters can be passed differently between each. However, whatever default shell you are dumped into should be sufficient for running the commands we are going to cover.

If your Linux boot distribution starts up in a GUI such as KDE or Gnome, you will want to look around for the icon to open up a shell console (typically, this looks like a sea shell).

Once you are at your console prompt, you will see either a prompt that looks like “#”, or maybe “machinename%”, or something to that affect.

If you are not logged in as root, which is the administrator account for Linux, type “su” at the prompt to switch over to the root user. This will enable full access privileges for the commands you want to execute over the system.

When performing any computer analysis, it is important you create a directory somewhere independent of the media you are examining to store your analysis results. In Linux, the `mkdir` command is used to create a new directory. Lets create a new evidence folder right off of the root of the computer system:

```
# mkdir /evidence
```

III. Examining Data

Floppy Disk Analysis

Now, lets say you received a floppy disk as apart of a criminal investigation into a drug dealer. You have no idea of the exact source computer of the floppy disk.

You have been requested to examine the floppy, see if you can find some incriminating evidence, and maybe even tie it specifically back to a computer.

The first thing you would do is create a backup image of the floppy disk. As the floppy disk is a piece of evidence, we want to preserve it in its originality the best we can. This image of the floppy disk will be an exact byte-for-byte replica of the floppy, copied to a special file on our hard disk.

In Windows, you will click on “A:” or “B:” to get to the floppy disk drive. In Linux, physical devices are treated as a file. In most cases, the floppy drive is `/dev/fd0`. If you had another floppy drive, it would most likely be `/dev/fd1`, and so forth.

Before inserting the floppy into our computer, check whether or not the switch on the floppy disk is set to be write protected or not. If it is not, then change it to a write protected state to ensure that no data can be modified on this original source disk. Note this in your Chain of Custody log that you should be keeping.

Next, one would insert the floppy disk into the drive. To see if you can view the contents of the floppy disk:

```
# ls /dev/fd0
```

This will bring up a brief listing of the contents on the disk, if it is in fact readable.

Next, you will want to proceed with making the image of the floppy. We will use a very helpful utility called `dd`. With this utility, you can specify an input file, an output file, and any other special parameters to replicate. In our case, the input file would be the floppy drive, and the output will be a file called `floppy1.img` within our `/evidence` folder.

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

The `bs` specifies the block size, which is 512k in this case. When copying a floppy, it is best to set this at 512. When copying such devices as hard drives, you may want to set this larger, such as 1k. The block size largely affects how quickly, or slowly, the system will copy the data. When in doubt, it is safer to go with a smaller number.

Now, let's obtain a unique signature called an SHA hash of the floppy disk. An SHA hash is a special cryptography algorithm. A brief synopsis of this is that no two files or disks can have the same SHA hash value if any bytes of data at all differ between them. This is extremely powerful in demonstrating in court the

integrity of a file or files. So, to obtain the SHA hash of the floppy disk drive /dev/fd0:

```
# sha1sum /dev/fd0
c3dae2b3b034a0d63a328b84c66d444103d76b2b /dev/fd0
```

That long string that starts with “c3dae...” is the returned hash value. You will want to save this information.

Now, lets obtain the hash value of the image file we created. As dd tries to perform an exact byte-for-byte replica of its source, we should expect the SHA hash to be the same:

```
# sha1sum /evidence/image.floppy1
c3dae2b3b034a0d63a328b84c66d444103d76b2b /evidence/image.floppy1
```

Notice the SHA hash is the exact same. This is very good! We have an exact replica of the evidence. You can remove the floppy and put it somewhere safe. Now, lets output this long hash to a file in our evidence folder. This will allow us to demonstrate to a court that our analysis was performed on an exact copy of the evidence and that it was not tampered with.

```
# sha1sum /evidence/floppy1.img >/evidence/floppy1img.sha1sum
```

To see the file you just created:

```
# cat /evidence/floppy1img.sha1sum
c3dae2b3b034a0d63a328b84c66d444103d76b2b /evidence/image.floppy1
```

Next, lets take advantage of a helpful utility called the “file” utility. This program takes a look at a particular file specified, and will try and look for unique “header” information to identify exactly what type of file it is. You typically expect a file that ends in .gif, such as farm.gif, to be a picture file. Likewise for .jpg files, .bmp files, etc. However, this may not be the case. You may call a file anything you would like. I can create a document within Microsoft Word, and save it as champlain.gif. This does not make it an image file. When encountering a file of unknown origin, this is a great place to start trying to figure out what was used to actually create the file.

So, lets try and use the “file” utility on our image itself, to see if we can identify the file system that is present.

```
# file /evidence/floppy1.img
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)
```

We can see it is an unlabelled floppy disk with the FAT file system. It appears to be in “MSDOS5.0” format, which tells us it was probably from a Microsoft

operating system. Linux natively supports reading the FAT file system, so we can actually read this disk.

The next step to actually reading files from the image is to do something known as mounting. When you mount a device in Linux, you are basically opening it up for access. When you first boot your Linux computer, the system automatically mounts your hard drive, CD-ROM drive, and any other disks that are connected. In our case here, we are going to be mounting the image file itself, and tricking the operating system into thinking this is a floppy disk. (2)

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

The first line creates a directory called analysis. Remember, Linux looks at devices at a file level. We will use this directory to attach the image we are mounting to. The next line tells us to mount the file system as read-only and not allow execution of binaries that are in the image. Additionally, notice the loop parameter. This is a special method within Linux to mount the image file using a loop back interface, which is where it tricks it into thinking it is a real floppy disk.

Now, lets go into our newly mounted image and take a look around. To do so, we can type:

```
# cd /mnt/analysis
```

and be within the mounted image.

The next step is to obtain an SHA hash of every single file on the floppy disk. A defense lawyer may later question whether the data has been altered since it was received, or at any time between when it was received and trial. You will want to demonstrate that without a doubt it has not. We will output this list to a file called floppy1img.sha1filehash within our evidence folder.

```
# find . -type f -exec sha1sum {} \;
>/evidence/floppy1img.sha1filehash
```

The above command uses the find utility. This will chain the name of each file it finds and send it to sha1sum, which is what the {} indicate. Next, the “>” tells Linux to output all of this information to a file specified after it as /evidence/floppy1img.sha1filehash.

Now, lets take a look at a friendly list of all the files on the disk itself. We can use the “ls” command to list the contents on the floppy. The “-alR” parameters are passed to the ls command and basically tell it to list all files, including dates, times, and permissions, and recursively go through the entire file system.

```
# cd /mnt/analysis
```

```
# ls -alR
.:
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. In our particular example above, we see an interesting file called crackdealers.d. However, when you simply try and view crackdealers.d with the cat command (`cat crackdealers.d`), you receive a bunch of gibberish in return. Lets see if the Linux file utility can identify what type of file this actually is.

```
# 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 of the raw text within the file to reveal more clues. To do so, we will use a utility called "strings". This utility basically goes inside of the specified file, and pulls out any raw text that it can recognize. Since this returned information may scroll past a single screen, we will also use a utility called "less". This handy utility will display given text one screen at a time. It allows you to use your keyboard arrow keys to scroll back and forth within the text. The "|" command basically tells Linux to take the results of the strings utility, and pipe this into the less utility.

```
# 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 the document 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. To do so, lets build a list of search words we want to seek out and put these all into a file. First, to do so, lets open a new text file with the vi text editor.

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

You will now be in what is infamously known as one of the most complex text editors in the world. Entire books are dedicated on how to use this monstrosity, so please bear with me. We will just be using it for a basic text file creation. When inside, press the Escape button, and then the letter “i” to insert new text. We can then type the following:

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

Note that there is no empty line after the word “Revenue” above. Now, to save it, press the Escape button, then type “:wq!”. Vi will read the wq!” commands which tell it to write the file (w), and quit (q). The ! tells it to do this right now.

To make sure it was created correctly:

```
# cat /evidence/searchcriteria.txt
Crack
617-515-5555
Ted
603-515-5555
NH
Revenue
#
```

Now, lets search the image for any occurrences of those words you typed in the /evidence/searchcriteria.txt file. To do so, we are going to use the grep utility. This is a very powerful text search utility. the parameters -aibf passed to the utility basically tell it to recursively search the destination, ignore binary data (b), and to use the input file (f) /evidence/searchcriteria.txt as our word list. We are going to be searching the floppy1.img file that we created earlier, which again, is an exact replica of our evidence. Any results will be outputted to the /evidence/searchcriteria.results file which we can review later.

```
# grep -aibf /evidence/searchcriteria.txt /evidence/floppy1.img
>/evidence/searchcriteria.results
```

Now, lets view our results file with the cat utility.

```
# cat /evidence/searchcriteria.results
[gibberish]
```

The cat utility will return our results, but with the embedded binary also. Lets see if we can view it in a bit friendlier format with the strings program:

```
# strings /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. The number at the beginning of each line tells us the byte location on the disk of roughly where the match was found. So, let's browse out to that area of the image and see what we can find. To do so, we are going to use what is known as a HEX viewing utility called xxd. This utility will display the raw binary and hexadecimal data written to a file. This is very powerful, in that it will show you all the raw characteristics of a file, even things that someone may be trying to hide.

```
# xxd -s 94322 /evidence/floppy1.img | less
000476f: 0000 0080 0100 0000 0000 0000 0000 0000 0000 .....
000477f: 0000 0080 0100 0000 0000 00d1 0100 0030 .....0
000478f: 0000 0001 0200 0000 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 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 as 11 Clarendon Apt 6 in Back Bay area of Boston.

Next, we may copy the crackdealers.d file to a machine capable of reading Microsoft Word files to read its interpreted contents as they may originally have been written. You may find under the document properties the "NH Department of Revenue" filled in. You may also try and use these utilities mentioned above against the other files on the floppy disk that we saw. For example, you may run "file" against the "addressbook" file. Is it a Microsoft Outlook address book? Is it just a Microsoft Excel spreadsheet listing? Maybe it's just a text file typed in notepad. The file utility will help try and narrow this down for you. And, even if it cannot figure out what type of file it is, you may use the xxd utility against the addressbook file itself to view or search its raw information.

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 will not get into how to open your computer and physically connect the drive, jumper settings, or anything of that nature. I would also 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.

First, you will want to see if we can identify any file partitions on the disk itself. A partition is basically a section of disk. Think of a disk as a pie, and a partition as a slice of pie. On your Windows computer at home, you may have a C:\ and D:\ hard disk drive letter. This does not necessarily mean you have two physical hard drives inside of the computer. Rather, you could have one physical drive with these two partitions.

So, lets use the fdisk utility to list (-l) the partitions on the hard disk /dev/hdc. Remember, Linux accesses physical devices as files. In our test system, the attached hard disk that we are analyzing may not necessarily automatically mount upon boot. However, this does not mean that Linux cannot find the drive sitting out there. In this example case, the physical disk is /dev/hdc. (3)

```
# 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 /dev/hdc disk has many different partitions. This includes some Windows FAT32 partitions, as well as some Linux partitions. 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 (had both Windows and Linux installed at the same time), or maybe had both operating systems installed at different points during its lifetime.

We should probably document this information in a file.

```
# fdisk -l /dev/hdc >/evidence/hdc.contents
```

After this has been completed, lets obtain the SHA hash of the partition we imaged and note this for comparison against our soon to be made image. We want to be sure this is an exact replica, and be able to provide this proof to a court upon request. Lets save this to the file hdc3.shalsum within our evidence folder.

```
# shalsum /dev/hdc3 >/evidence/hdc3.shalsum
```

Next, as we did with the floppy, lets create an image of the partition from the disk. We will again use the powerful dd utility, and use the Linux swap slice as our input file. We will use a block size of 1k, which is a good safe number when using dd against a hard disk drive. Further, we will be passing the parameters sync, which tells dd to keep the data synchronized in exact location between the input and output, and noerror, which tells dd to “pad” any error spots it tries to read from the input file. When using a computer hard disk, over time small bits of it go bad. As this happens, the operating system will mark this area as bad and skip using it in the future. However, dd does not care about this, and our command will tell it to even try and copy these bad parts. Ultimately, we will be outputting this image to the file hdc3.img within the /evidence folder.

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

Now lets obtain the SHA hash of our image file itself, and save it to hdc3img.shalsum within our evidence folder.

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

Now, lets view the SHA hash files we created for both the partition and the image to ensure that they match:

```
# cat /evidence/hdc3.shalsum  
f36d043671f59ebe79a8c347132e62bbb42ab45f /dev/hdc3  
# cat /evidence/hdc3img.shalsum  
f36d043671f59ebe79a8c347132e62bbb42ab45f /evidence/hdc3.img
```

Excellent, they match! Next, we can use the file utility to see if we can obtain further information about the file system within the image.

```
# 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.

Next, as we did with the floppy disk image, you would probably mount the file system. However, in the above specific example, you cannot actually mount a Linux swap space partition. However, for other file system types, of which Linux can recognize more than 30 different types, you could. Further, you can still use `grep`, `xxd`, and other utilities against the image file itself, regardless of the file system.

Here is an example of mounting the file system, this time not specifying what type of file system. When you do not specify the file system, `mount` will make a best guess as to what it might be. This is often sufficient.

```
# mount -o ro,noexec,loop /evidence/hdc3.img /mnt/analysis
```

As with the floppy, your analysis and tools used may follow a similar pattern from herein. 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.

IV. Conclusion

You should now have an introductory familiarity with some basic steps to performing 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. Burdach, M., *Digital forensics of the physical memory*, http://forensic.seccure.net/pdf/mburdach_digital_forensics_of_physical_memory.pdf
5. Nemeth, E., Snyder, G., Seebass, S., Hein, T. *UNIX System Administration Handbook*, 2nd ed, Prentice Hall PTR, Upper Saddle River, New Jersey, 1995.