Forensic Disk Imaging Step-By-Step

By Synjunkie

For one reason or another you may want to make a copy of a hard disk. I will describe methods to create a bit-for-bit copy of a hard disk either to a local device or over a network.

The thing to remember throughout the examples listed below is Linux thinks of everything as a file. So the file it sees as hda in the /dev directory is actually the harddisk. The following software will be used in the examples listed below:

- A bootable live linux distro that does not auto mount drives such as Helix
- dd
- nc
- split
- md5sum
- cat

dd, nc, md5sum, cat and split are available on Linux and Windows.

Regarding hardware you will require the following:

- 2 x Computers (if creating a copy across a network)
- USB thumb drive
- USB hard drive (If creating the image to a USB hard drive)

Example 1 – A Copy Across A Network

To make a copy across a network you will need 2 computers, the target computer, Computer01, and the computer you will be copying to, Computer02.

1. Insert the Linux boot disk into Computer01 and boot the system into Linux.

2. Insert the USB thumb drive, if this doesn't automatically mount it will require mounting. In my examples below I will assume it is /dev/sdb1 and has been mounted as /media/USB.

3. Locate the disk you want to copy in the /dev directory, in my examples the hard disk will be called hda yours maybe something similar.

4. Using the command md5sum /dev/hda >/mount/USB/diskimage_md5hash.txt create a MD5 hash of the drive on the mounted USB drive so you can test this against the copied file to verify the integrity.

5. On Computer02 make sure you have enough diskspace to accommodate a file the size of the disk you are going to copy and using netcat (nc) run the command

   nc –L –p 6677 >c:\diskimage.img

   What you have done here is to set up netcat (nc) to listen persistently (-L) on port 6677 (-p 6677) and send the output to a file on C:\ of Computer02 (>c:\diskimage.img).

6. From Computer01 run the following command:

dd if=/dev/hda | nc 192.168.1.2 6677

This command assumes that the IP address of Computer02 is 192.168.1.2. By running this command you will be copying the input file /dev/hda (if=/dev/hda) from Computer01 to C:\diskimage.img on Computer02 using netcat (nc).

7. Finally, after the copy has finished you can run md5sum on Computer02 against the C:\diskimage.img file on Computer02 and compare this to the md5sum taken earlier to verify the copies are identical.

Example 2 – A Local Copy to a USB Storage Device

In this example you will need only the Target PC and a USB storage device large enough to hold the image.

1. Insert the Linux boot disk into the computer and boot the system into Linux.

2. Connect the USB storage device, if this doesn't automatically mount it will require mounting. In my examples below I will assume it is /dev/sdb1 and has been mounted as /media/USB.

3. Locate the disk you want to copy in the /dev directory, in my examples the hard disk will be called hda yours maybe something similar.

4. Using the command md5sum /dev/hda >/mount/USB/diskimage_md5hash.txt create a MD5 hash of the drive on the mounted USB device so you can test this against the copied file to verify the integrity.

5. Run the following command:

   dd if=/dev/hda of=/media/usb/diskimage.img

   This will copy the disk as a file onto the USB storage device as diskimage.img.

6. Create another md5 hash of the image on the storage device and compare to the original to verify the integrity of the copy.

The result of both of the examples above is a forensically sound image of the hard disk.

Advanced Usage of dd for Imaging

Whilst using the methods above you may come across issues. For example, if the PC cannot read some of the sectors of the drive you are copying, or if the file needs splitting to fit onto CD's. Or if the image needs slitting to fit on a device that is FAT32 and requires files to be smaller than 2GB.

Copying an image from a disk with bad sectors

When imaging a drive that is starting to have some bad sectors the command below can be used:

dd if=/dev/hda of=/media/USB conv=noerror,sync

This will allow dd to proceed past read errors, and pad the destination with 0's where there were errors on the source drive (so your size and offsets will match). If you do this, you may want to consider redirecting standard-error out to a file, so you have a record of where your errors were.

Splitting images

This can be done using a couple of different methods. The easiest method is by using the split program. The syntax for the command if you required a 4GB image to fit on CD's would be:

dd if=/dev/hda | split –b 620m - /USB/sda/

This will run the input file (/dev/hda) through split and create several files of 620MB (-b 620m) in the directory /USB/sda/. The files will usually be called x** (* denotes a wildcard in this example). These files can be reformed into an image file using the cat command.

Cat x* > bigimage.img

Then create a hash of the file using md5sum and compare to the original hash value.

Md5sum bigimage.img

Alternatively, if split is not available you can use dd by itself but use the skip, bs (block size) and count switches to prevent it from reading from the beginning of the file.

dd if=dev/hda of=/media/USB/image1.img bs=1M count=620
dd if=dev/hda of=/media/USB/image2.img bs=1M count=620 skip= 621
dd if=dev/hda of=/media/USB/image3.img bs=1M count=620 skip= 1241
dd if=dev/hda of=/media/USB/image4.img bs=1M count=620 skip= 1861
dd if=dev/hda of=/media/USB/image5.img bs=1M count=620 skip= 2481

etc……until the end of the input file.

What is happening here is you are telling dd to work in 1MB blocks (bs=1M), to only copy 620MB at a time (count=620) and in some cases to skip to a particular part of the input file (skip=621 etc…), thus creating several images that can then be copied to CD's. Once on the target system and in the same directory (I will assume directory is /home/me) they can be put back together into a single image using the command below:

Cat /home/me/image* > bigimage.img

Md5sum can be run against this image and compared to the original md5 hash to verify the integrity.

Dd To a Zipped Image

You can pipe dd through gzip to save on some disk space.

dd if=/dev/hda | gzip -f > /media/USB/compressed_image.img.gz

Forensic Disk Imaging Step-By-Step
By Synjunkie

Using Split & Gzip Together

To help cope with size limits both gzip and split can be used together. This has the benefit of splitting the image and zipping it up also to save space and requires less work. Below is the syntax used to perform this and an explanation of the command:

dd if=/dev/hda | gzip –c | split -b 2000m - /media/USB/image.img.gz.

1. dd is used to take an image of the harddrive.
2. This is passed to gzip (-c is to stdout)
3. The compressed image is then piped to the split tool (split then creates the files image.img.gzaa, image.img.gzab, etc).

To restore the multi-file backup, run the command below:

cat /USB/image.img.gz* | gzip -dc | dd of=/dev/had

1. Cat displays the contents of the zipped and split image files to stdout in order.
2. Results are piped through gzip and decompressed.
3. And are then written to the hard drive with dd.

Creating empty disk images

To create an empty disk image, get the data from /dev/zero. To create a 10MB image or file:

dd if=/dev/zero of=image bs=1M count=1024

Or

dd of=image bs=1M count=0 seek=1024

In the second example nothing is written, not even zeroes, we just seek 10MB into the file and close it. The result is a sparse file that is implicitly full of 10MB of zeroes, but that takes no disk space. ls -la will show 10MB, both du and df will show 0. When the file is written to, Linux will allocate disk space for the data. ls will continue to show 10MB, but du will gradually approach 10MB.

Notes:

Whilst researching the use of dd another tool was brought to my attention which is called dcfldd. This tool is like dd in many ways and uses similar syntax but is also able to produce hashes on the fly and can provide status of copying files amongst other useful features. It's available on both Linux and Windows.

Thanks to the guys at BinRev for ideas whilst creating this entry.