The $LogFile and Device Alteration: An experiment

By Nanni Bassetti – ISSA member, Italy Chapter

By this experiment we can affirm that if a computer investigator connects an NTFS drive for duplication without using a write blocker and without browsing it, the drive will be modified, but no one could prove it because there is no apparent evidence of this changing.

Abstract

In this experiment an NTFS USB device was attached to a computer running Windows OS and then detached. The hash code of the device was calculated before and after changes, but there is no evidence expressed in date and time of this changing, when it happened? We saw that the alteration was located in the *$LogFile*, a special file of the MFT (Master File Table), but we saw that there was not any change in the date and time of this special file. By this experiment we can affirm that if a computer investigator connects an NTFS drive for duplication without using a write blocker and without browsing it, the drive will be modified, but no one could prove it because there is no apparent evidence of this changing.

The best practices of the computer forensics encourage the operators to use the write blockers to avoid the alteration of the seized supports. This is a good practice, because some operators could browse the seized device before acquiring it and so altering many access dates and times. But, what happens if we attach directly to an USB port an NTFS device?

This is an experiment to understand what changes when an USB device is attached to a Windows OS system and soon detached, seeing if it is possible to discover an eventual alteration by the changing of the date and time of something in the device file system.

We will use an empty NTFS pendrive and will attach it to a computer running Windows XP by the USB port. We will notice that the *$Logfile* will change, but not its metadata (date and time). In this experiment we used The Sleuth Kit (TSK) 3.0.1,[1] DD, MD5SUM, and HEXCMP2.[2]

1 http://www.sleuthkit.org.

2 http://www.softempire.com/hexcmp2.html.

The experiment

1) Create a bitstream image by DD (pen1.dd) of a NOT mounted, empty NTFS pendrive using a computer running Linux OS modified for computer forensics. This means that there should not be any alteration made to the device by the operating system because the pendrive is not mounted:

```
dd if=/dev/sdb of=/media/pen1/pen1.dd
```

2) Calculate the MD5 hash code of pen1.dd:

```
md5sum /media/pen1/pen1.dd
62650F196045DA49FF9EF8AC1390E6D2
```

3) Attach the empty NTFS pendrive to a computer running Windows XP system.

4) Detach it using the SAFE DISCONNECTION procedure.

5) Create another bitstream image by DD (pen2.dd) of the NOT mounted, empty NTFS pendrive, using the Linux box as before:

```
dd if=/dev/sdb of=/media/pen2/pen2.dd
```

6) Calculate the MD5 hash code of pen2.dd:

```
md5sum /media/pen2/pen2.dd
29A203D4CD5BA8D1C6A67CFE11D98608
```

7) Compare the two hash codes: MD5 hash code of the `pen1.dd` is not equal to MD5 hash code of the `pen2.dd`. This means there has been an alteration by simply connecting and disconnecting an empty NTFS device to a Windows system by the USB port.

```
62650F196045DA49FF9EF8AC1390E6D2 does not equal
29A203D4CD5BA8D1C6A67CFE11D98608
```

Using HexCMP2 Demo for Windows (Figure 1), we compared the two images files (pen1.dd and pen2.dd) and we found all the differences were in the $LogFile cluster, that is the journal of the NTFS file system.

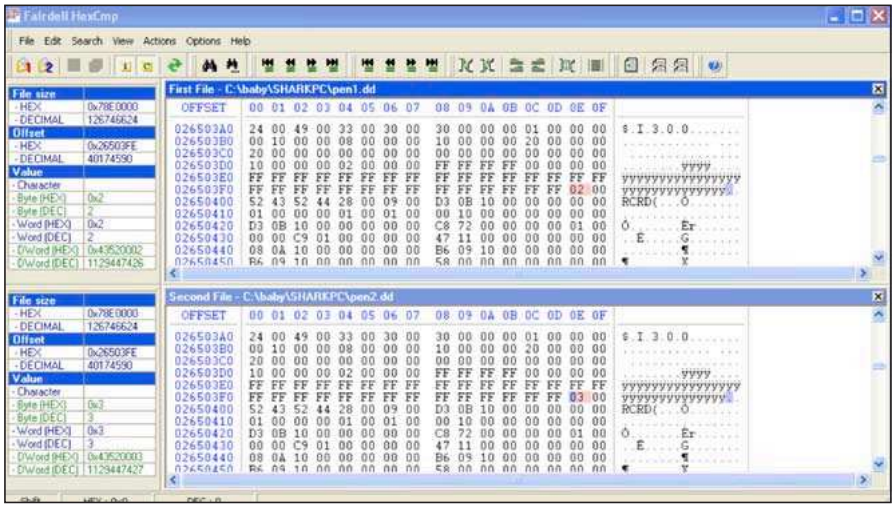

Figure 1 – HexCMP2 result showing one change between the two images

Now, let's discover what has changed. For this purpose we use TSK *mmls* to see the partition starting sector and TSK *fsstat* to see the cluster size. For instance, the last difference is in the decimal offset: 40203262

```
mmls pen1.dd
```

This determines the partition starting sector: 32

```
fsstat -f ntfs -o 32 pen1.dd
```

The cluster size: 512

The operation 40203262 / 512 = 78521 represents the offset expressed in sectors.

The following tools are part of TSK:

Using IFIND we can retrieve the i-node pointing to the cluster number found by HexCmp2:

```
ifind -f ntfs -o 32 -d 78521 pen2.dd
```

i-node: 2-128-1

Using FFIND we can find the file corresponding to the i-node 2-128-1:

```
ffind -f ntfs -o 32 pen2.dd 2-128-1
```

file: //$LogFile

Using ISTAT we can see the $LogFile metadata:

```
istat -f ntfs -o 32 pen1.dd 2-128-1 | less
```

<...>
$FILE_NAME Attribute Values:
Flags: Hidden, System
Name: $LogFile
Parent MFT Entry: 5 Sequence: 5
Allocated Size: 2097152
Actual Size: 2097152
Created: Tue Mar 18 15:05:19 2009
File Modified: Tue Mar 18 15:05:19 2009
MFT Modified: Tue Mar 18 15:05:19 2009
Accessed: Tue Mar 18 15:05:19 2009

```
istat -f ntfs -o 32 pen2.dd 2-128-1 | less
```

<...>
$FILE_NAME Attribute Values:
Flags: Hidden, System
Name: $LogFile
Parent MFT Entry: 5 Sequence: 5
Allocated Size: 2097152
Actual Size: 2097152
Created: Tue Mar 18 15:05:19 2009
File Modified: Tue Mar 18 15:05:19 2009
MFT Modified: Tue Mar 18 15:05:19 2009
Accessed: Tue Mar 18 15:05:19 2009

If we do all the procedure without using the SAFE DISCONNECTION, but brutally detach the pendrive, we obtain the same results, though the last change is located at the decimal offset 40174590. Doing 40174590 / 512 = 78465, we notice that there are less alterations, because 78465 < 78521, because the SAFE DISCONNECTION procedure writes more data than the brutal disconnection.

The date and time of the $LogFile, into the two image files, are equal, so this is my question:

Why does the $LogFile change but not its metadata?

In this way, someone could acquire a device by Windows without using a write blocker and he could calculate the hash, altering the seized device. In a different time, another consultant will acquire the seized device, using a write blocker and the hash code will be equal to the first hash code, but we know that the original one has been altered in the $Logfile, but we have not the date nor the time evidence of the alteration.

Conclusion

We don't know why the date and time of the $LogFile do not change, even if the $LogFile changes – maybe because the NTFS is not an open source project and many specifications are still hidden.

This is only an academic, though very interesting, question, because pratically the alterations do not involve the files and their metadata hosted on the device (we are simply attaching and detaching without browsing the device) but involve only the $LogFile. But from a scientific point of view, we can say that the copy is not equal to the original.

About the Author

Nanni Bassetti has a Computer Science degree from the University of Bari (Italy) and works in the Web development, security and computer forensics. He is member of the Caine development and testing team and creator of Computer Forensics Italy (http://www.cfitaly.net), a great online community and mailing list. He may be reached at http://www.nannibassetti.com or digitfor@gmail.com.