

Welcome to MSFN Forum

Register now to gain access to all of our features. Once registered and logged in, you will be able to create topics, post replies to existing threads, give reputation to your fellow members, get your own private messenger, post status updates, manage your profile and so much more. This message will be removed once you have signed in.

Login to Account

Create an Account

- App Management
 - Patch Management
 - Policy Management
 - Software Deployment

Download



Search



NTFS File Lister using \$MFT

Started by CharlotteTheHarlot , Oct 17 2012 02:52 AM

CharlotteTheHarlot Posted 17 October 2012 - 02:52 AM

-(As there is no forum just for NTFS and I needed to choose one, WinXP is as good as any. If there is a better choice please let me know and I'll ask the mods to move this.)-

Background ... For some time this has been like a quest for the holy grail - obtaining a perfect file list off an NTFS volume. Specifically a tool that directly accesses the \$metadata, bypassing NTFS and Windows API functions. Doing this allows the utility to list or copy any file on an NTFS volume, including those that might be in use or locked down or inaccessible from tangled ACL permissions. Consider a volume that is not in use, perhaps it is slaved in another computer for forensic analysis, accessing files through the \$MFT (Master File Table) and other related metadata will allow access to all files not just those that are "allowed" by the host operating system which you are using to view this volume. Ideally, such a tool should be able to directly process the metadata like the \$MFT even when it is copied to as a discrete file (i.e., the ability to select an \$MFT file to process). So if you copy the \$MFT as a file onto some other disk, you can then peruse it at your leisure, list it, extract data, etc. Finally, there is the added benefit that this method is much faster since it lacks the overhead found in the standard file system sequential method.

What Am I Looking For? ... The thing I have personally most sought (and the topic of this thread) is a replacement DIR lister, one which does so using the \$MFT and allows specific fields to be exported to or echoed as plain text (like DIR) or if necessary CSV (which unfortunately would require extra post processing). I have found a few programs that come close but so far none have all the requirements. In summary, I would like to be able to save a perfect file list of an NTFS volume with the following fields ...

- **Modified Date/Time** (YYYY-MM-DD HH:MM:SS.xxx) converted to local!
- **Creation Date/Time** (YYYY-MM-DD HH:MM:SS.xxx) converted to local!
- **Accessed Date/Time** (YYYY-MM-DD HH:MM:SS.xxx) converted to local!
- **Size** (x,xxx) with commas!
- **FilePath\FileName**

... displayed something like this ...

Modified	Created	Accessed	Size	Name
YYYY-MM-DD HH:MM:SS.xxx	YYYY-MM-DD HH:MM:SS.xxx	YYYY-MM-DD HH:MM:SS.xxx	x,xxx,xxx,xxx	C:\Tem
YYYY-MM-DD HH:MM:SS.xxx	YYYY-MM-DD HH:MM:SS.xxx	YYYY-MM-DD HH:MM:SS.xxx	x,xxx,xxx,xxx	C:\Real

Why Do This? ... Such an output could then be sorted by columns: **1**, **26**, or **51** resulting in a log sorted by **Modified**, **Created**, or **Accessed**. Coupled with other data pulled from the NTFS journal and elsewhere, a very thorough forensic history can be utilized for running down malware and other problems, especially if this procedure is performed periodically and you have different logs to compare. "Sorting" in itself is NOT a requirement of a perfect DIR lister, in fact I would say that it is not desirable at all, because it is easily done post-process by dedicated sort utilities or a good editor like UltraEdit. "Formatting" of the data into a consistent output with predictable columns absolutely **is** a requirement. The ordering of those fields is also NOT critically important since you can easily sort by any particular column, however the path\filename absolutely **MUST** be the last field since it is the only one that has unknown total column width. The first three (or six counting "time" separately) are obviously fixed-width. For the file size column we always know the maximum possible size (NTFS is 256 TB last time I checked) but it can be assumed that the largest file will not be larger than the volume it resides on, so for example that field could be set to x,xxx,xxx,xxx,xxx maximum if the volume was 2TB. As a matter of fact, the first two examples below using the DIR command do exactly this because the padding shown allows for precisely x,xxx,xxx,x19,761 bytes.

There are obviously many more possible output fields that could be extracted from the \$MFT and saved to a drive file list, but I would like to confine them to only these few so that a log file would be more manageable when listing every single file present on a large volume. I should note that some of the utilities I have tested also can log slack space and deleted files, and other things that are important in forensic analysis. For example they can pull in and display file attributes and owners, but this means that extra processing occurs (such as reading from other \$metadata and specific file streams) which will add to the overhead resulting in a longer time to finish making the list. So I figure that the utility already has enough to do, decoding each \$MFT entry and output formatting when it converts to local time, decimal, punctuating, justifying, etc. KISS theory, Keep It Simple Stupid.

In all examples below I have targeted one specific test file on a live NTFS system disk. Remote access to a volume, or parsing a standalone saved \$MFT has NOT been tested yet. **Important:** My computers are always set to **YYYY-MM-DD** and 24-hour time format. This is to reduce the incidents of encountering MM/DD/YYYY or even the utterly ridiculous MM/DD/YY in various logs and results files because so many programs simply output their results using the system default rather than implementing the more sortable and much more sane method of YYYY-MM-DD. If your system is set to something different I believe you can use the **DIR /4** parameter to adjust the year field but unless I am mistaken I believe there is nothing (except changing the system time format) that can fix the broken 12-hour time format which produces such insanity as 06:02 PM, or 12:59 AM followed immediately by 01:00 AM, or 11:59 PM followed immediately by 12:00 AM. These are sorting nightmares that make the time field practically useless and IMHO decreases the IQ of the human race in the process.

Here is the default output from DIR ...

EXAMPLE 01 ... DIR C:\Temp /a /s /x ... (CMD.EXE)

```
Volume in drive C is xxxxxxxxxxxx
Volume Serial Number is xxxx-xxxx

Directory of C:\Temp

2012-04-13 01:08      19,761      Test.gif
           1 File(s)      19,761 bytes
```

Aside from the fact that DIR is using Windows and NTFS API functions (access limitations can be in effect) and is very slow (especially on entire volumes, at least on the first pass through before the metadata is cached), however it does get part of the job done. With a little more development from Microsoft the DIR command could actually be quite useful, although certain files will never be displayed due to access limitations, so it will never be suitable for forensic use. **Shortcomings:** Well the time precision is too short (no milliseconds) and there is no option to list CREATED or ACCESSED date/times. But you can see that they aligned the output fields with sufficient padding for various file sizes so that the column widths are maintained.

NOTE: This example uses the **DIR /x** parameter which adds padding for an extra column to hold any 8.3 SFN (short filename) that may exist for a particular NTFS file.

EXAMPLE 02 ... DIR C:\Temp /a /s ... (CMD.EXE and COMMAND.COM)

Volume in drive C is xxxxxxxxxx
Volume Serial Number is xxxx-xxxx

Directory of C:\Temp

```
2012-04-13 01:08      19,761 Test.gif
               1 File(s)      19,761 bytes
```

Just for comparison sake, this is the output **without** the /x parameter so we see the missing field between size and filename.

NOTE: Using COMMAND.COM exactly matches what CMD.EXE shows when the **DIR /x** parameter is left off.

EXAMPLE 3 ... NtfsDir.exe C:\Temp -recurse 9999 >Output.txt

ntfsdir ver: 1.01, Copyright (c) TZWorks LLC
cmd: NtfsDir.exe C:\Temp -recurse 9999

last modified [UTC]	MFT & seq num	size	name
04/13/2012 05:08:14.203	484565 91	20480	C:\Temp\Test.gif

A promising 3rd party utility, **NTFSDIR** (which does use \$MFT and not the file system) creates output that is somewhat similar to DIR, and displays the necessary milliseconds for the time precision. **Shortcomings:** Unfortunately it insists on using the MM/DD/YYYY date method (but at least it is 4-digit). This is not a deal-breaker though because it is still sortable using a good editor that allows 'key' column preferences (in this case column 7) but that adds extra processing later. The time is left in UTC time format without conversion to local and although this will not affect sorting, it is a PITA when switching back and forth between a file list and a live system on local time. Another failure is the lack of commas for file sizes which also does not impact on sorting of course, but it is a cosmetic no-no that slows down interpretation of long numbers and therefore can encourage human errors. Finally, like DIR, there is no CREATED or ACCESSED date/times but instead it adds two \$MFT specific fields of no particular value in this case.

NOTE-1: The parameter **-recurse 9999** (equivalent to **DIR /s**) is a kludge because unlike some of their utilities, this one does not have a **-recurse 0** to signal infinite subdirectory recursion! 9999 is an arbitrary number I chose. This kludge is present in all examples that follow that have this bug or oversight.

NOTE-2: IMPORTANT! there is a quirk or bug in NTFSDIR (and some of their other utilities) which inexplicably breaks scripting in a batch file for more than one command. In other words, if you place two consecutive NTFSDIR commands on different lines, only the first one will execute, after that the batch file terminates and the command windows closes!

NOTE-3: The redirect parameter **>Output.txt** is not shown there in the output header but it was definitely included for the test to get the output into a file.

EXAMPLE 4 ... NtfsDir.exe C:\Temp -recurse 9999 -csv >Output.txt

ntfsdir ver: 1.01, Copyright (c) TZWorks LLC
cmd: NtfsDir.exe C:\Temp -recurse 9999 -csv
type, filename, symbolic link, size, attributes, MFT entry, sequence #, flags, cdate (stdinfo), ctime (stdinfo), mdate (stdinfo), file, C:\Temp\Test.gif, , 0x000000005000, 0x00002020, 0x0000000764d5, 0x005b, 0x0001, 04/13/2012, 05:08:

Adding the **-csv** parameter to the previous example command results not only in CSV output (comma separated values), but it also yields some extra data fields. This one actually delivers all the required fields I want to see, albeit out of order.

Shortcomings: Unfortunately it puts the filename first (definite deal-breaker!), and it uses hex for file size with no option for decimal conversion. One GIGANTIC mistake here is their using the barest CSV delimiter possible ", "(comma,space) which is certainly not helpful on file systems that allow exactly those two consecutive characters in a filename! Ideally there should be a selection of delimiter choice (not specifically a criticism of these utilities, but any program that uses CSV or another output method). Again, the redirect parameter **>Output.txt** is not shown there in the header but it was definitely included for the test to get the output into a file. And once again, dates are using the MM/DD/YYYY format and times are left as UTC.

EXAMPLE 5 ... NtfsDir.exe C:\Temp -recurse 9999 -alldates -csv >Output.txt

```
ntfsdir ver: 1.01, Copyright (c) TZWorks LLC
cmd: NtfsDir.exe C:\Temp -recurse 9999 -csv
type, filename, symbolic link, size, attributes, MFT entry, sequence #, flags, cdate (stdinfo), ctime (stdinfo), mdate (stdinfo)
file, C:\Temp\Test.gif, , 0x000000005000, 0x00002020, 0x00000000764d5, 0x005b, 0x0001, 04/13/2012, 05:08:
```

This is the same as the previous command, with **-alldates** added. As such, it also delivers all the required fields plus many more, with no ability to toggle any specific fields off. **Shortcomings:** As previously shown, it puts the filename first, it uses hex for everything with no option for decimal, dates in MM/DD/YYYY, time in UTC. Also, the mistake is repeated here is using the CSV delimiter ", "(comma,space). And once again, the redirect parameter **>Output.txt** is not shown there in the header but it was definitely included for the test to get the output into a file.

NOTE-1: That command line actually has the parameter **-alldates**, but the entry in the header for some reason does not show it (some bug?) nor does it ever seem to appear in any testing.

NOTE-2: Confusingly, I have learned that contrary to the **-?** usage information, the parameter **-csv** alone returns some of the information that **-alldates** does. So, comparing examples 3 (no switch), 4 (**-csv**), and 5 (**-alldates -csv**) return an increasing amount of information for each file. The **-csv** switch does NOT simply convert output to CSV, it in fact adds more data as well.

EXAMPLE 6 ... NtfsDir.exe C:\Temp -recurse 9999 -xml Output.txt

```
<?xml version="1.0" encoding="utf-8"?>
<logdata starting_path="C:\Temp" date="10/15/2012" time="04:06:08.531">
<data_entry><directory>C:\Temp\</directory>
<filename>Test.gif</filename><dir/><size>20480</size><sym_link/><file_create_date>04/13/2012</file
```

This is the same as example 4 but now using **-xml** instead of **-csv**. **Shortcomings:** Most of the same, MM/DD/YYYY, UTC, no commas, but at least the hex2dec was accomplished. This XML output has a peculiarity of breaking the path\filename into two separate lines though which introduces yet more extra post-processing. Since all the other XML fields were concatenated onto one line, it seems strange to leave that one hanging there.

NOTE-1: Important to remember that the redirect symbol ">" is not required and causes this message: **Error: option requires an output file: -xml <output file>**, so it **must** be left off. (possible bug?)

NOTE-2: This XML output does not produce any output header echoing the command line and copyright.

EXAMPLE 7 ... NtfsDir.exe C:\Temp -recurse 9999 -alldates -xml Output.txt

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<logdata starting_path="C:\Temp" date="10/15/2012" time="04:09:10.284">

<data_entry><directory>C:\Temp\</directory>
<filename>Test.gif</filename><dir/><size>20480</size><sym_link/><file_create_date>04/13/2012</file
```

This is the same as example 5 but using **-xml** instead of **-csv**. It contains a few extra fields compared to the previous example 6 due to the additional parameter: **-alldates**. **Shortcomings:** All the previous shortcomings are present. Also note the lack of the redirect symbol ">" in the command line since it is using the **-xml** as explained above.

EXAMPLE 8 ... NtfsWalk.exe -partition "c" -filter_name "test.gif" -out Output.txt

```
cmdline: NtfsWalk.exe -partition "c" -filter_name "test.gif" -out Output.txt

mft entry | seq | parent mft | type | ext | ref | date | time [UTC] | MACB | other info | path and file
0x000764d5 | 91 | 0x00002e4c | file | gif | 1 | 04/13/2012 | 05:08:14.109 | si:[...b]; fn:[...b] | | [root]\
0x000764d5 | 91 | 0x00002e4c | file | gif | 1 | 04/13/2012 | 05:08:14.156 | fn:[mac.] | | [root]\T
0x000764d5 | 91 | 0x00002e4c | file | gif | 1 | 04/13/2012 | 05:08:14.203 | si:[ma..] | | [root]\T
0x000764d5 | 91 | 0x00002e4c | file | gif | 1 | 10/01/2012 | 23:45:50.062 | si:[..c.] | | [root]\Te
```

This 3rd party utility **NTFSWALK** (again from TzWorks) also uses direct access to the \$MFT metadata, but is drive-based rather than directory-based. In theory this is more appropriate for the task at hand which is listing the complete contents of a volume/drive. The output has a different style header only consisting of the executed command line. **Shortcomings:** What immediately prevents this tool as a file list solution is the fact that it breaks out each \$MFT entry into separate output lines for each so-called inode, effectively generating a much longer list since each file almost always has several. If you look closely you can see that the **four** output lines for this same test file represent in this order: **<file_create_time>**, **<name_altered_time>**, **<file_altered_time>** and **<file_mft_changed_time>**. Another fatal feature can be seen in the default width for the **"ext"** field is far too small and will grow or truncate (haven't tested yet) when gigantic file extensions are encountered (e.g., CLSID's). Even worse is the fact that an extra field is displayed past FilePath\FileName which is a no-no because there are sure to be gigantic paths and filenames on any modern Windows installation (.Net, SxS, etc) and that last field will certainly be pushed out of alignment (or the file name/path truncated) and both are absolute deal-breakers. Additionally, there is the MM/DD/YYYY dates and UTC times. The file time precision is perfect since milliseconds are included in the output. The data is displayed in hex, but unlike previous examples this can be toggled away with a switch. There is also that very minor cosmetic bug in the header row output (they just need to pad the word **"ext"** with an extra space) which causes the column vertical divider to be misaligned only on the first row.

NOTE: This tool, NTFSWALK ***is*** batch scriptable, in other words you can have more than one line using it and the batch file will proceed as expected, NOT terminating prematurely as it would with NTFSDIR.

EXAMPLE 9 ... NtfsWalk.exe -partition "c" -filter_name "test.gif" -base10 -out Output.txt

```
cmdline: NtfsWalk.exe -partition "c" -filter_name "test.gif" -base10 -out Output.txt

mft entry | seq | parent mft | type | ext | ref | date | time [UTC] | MACB | other info | path and file
484565 | 91 | 11852 | file | gif | 1 | 04/13/2012 | 05:08:14.109 | si:[...b]; fn:[...b] | | [root]\Temp\T
484565 | 91 | 11852 | file | gif | 1 | 04/13/2012 | 05:08:14.156 | fn:[mac.] | | [root]\Temp\Tes
484565 | 91 | 11852 | file | gif | 1 | 04/13/2012 | 05:08:14.203 | si:[ma..] | | [root]\Temp\Tes
484565 | 91 | 11852 | file | gif | 1 | 10/01/2012 | 23:45:50.062 | si:[..c.] | | [root]\Temp\Test.
```

The exact same command with the added **-base10** to switch the most of the hex to values decimal (no commas unfortunately). **Shortcomings:** All the other problems listed previously are present here.

EXAMPLE 10 ... NtfsWalk.exe -partition "c" -filter_name "test.gif" -csv -out Output.txt

```
cmdline: NtfsWalk.exe -partition "c" -filter_name "test.gif" -csv -out Output.txt
```

```
mft entry,seqnum,parent mft,type,ext,ref,date,time [UTC],MACB,other info,path and filename,various data types,
0x000764d5,91,0x00002e4c,file,gif, 1,04/13/2012, 05:08:14.109,si:[...b]; fn:[...b],[root]\Temp\Test.gif,[unnamed data : sz:
0x000764d5,91,0x00002e4c,file,gif, 1,04/13/2012, 05:08:14.156,fn:[mac.],[root]\Temp\Test.gif,[unnamed data : sz:
0x000764d5,91,0x00002e4c,file,gif, 1,04/13/2012, 05:08:14.203,si:[ma.],[root]\Temp\Test.gif,[unnamed data : sz:
0x000764d5,91,0x00002e4c,file,gif, 1,10/01/2012, 23:45:50.062,si:[.c.],[root]\Temp\Test.gif,[unnamed data : sz: 0
```

The same command as example 8 but adding the **-csv** switch. It gives us the exact same output with delimiters between the fields. However a GIGANTIC problem (or bug) surfaces here, the delimiter is sometimes "," (comma,space) and other times just "," (comma). Of course as mentioned earlier, commas are problematic when dealing with LFN's anyway . But a comma alone is even worse. **Shortcomings:** All the other problems listed previously are present here.

EXAMPLE 11 ... NtfsWalk.exe -partition "c" -filter_name "test.gif" -base10 -csv -out Output.txt

```
cmdline: NtfsWalk.exe -partition "c" -filter_name "test.gif" -base10 -csv -out Output.txt
```

```
mft entry,seqnum,parent mft,type,ext,ref,date,time [UTC],MACB,other info,path and filename,various data types,
484565,91,11852,file,gif,1,04/13/2012, 05:08:14.109,si:[...b]; fn:[...b],[root]\Temp\Test.gif,[unnamed data : sz: 0x4d3
484565,91,11852,file,gif,1,04/13/2012, 05:08:14.156,fn:[mac.],[root]\Temp\Test.gif,[unnamed data : sz: 0x4d31 ],
484565,91,11852,file,gif,1,04/13/2012, 05:08:14.203,si:[ma.],[root]\Temp\Test.gif,[unnamed data : sz: 0x4d31 ],
484565,91,11852,file,gif,1,10/01/2012, 23:45:50.062,si:[.c.],[root]\Temp\Test.gif,[unnamed data : sz: 0x4d31 ],
```

The same command as example 9 but adding the **-csv** switch. Because of the **-base10** switch in example 9 and this one, most of the hex values are now decimal. Previous comment in example 10 applies to this one as well.

EXAMPLE 12 ... Wisp.exe -partition c -path c:\temp -level 9999 -base10 -all >Output.txt

```
Name:      c:\temp\Test.gif
MFT Entry:  0x0000000764d5 [484565]
MFT Sequence:  0x005b [91]
Parent Entry:  0x00000002e4c [11852]
Parent Sequence:  0x0001 [1]
Entry:      valid
Modify time [utc]:  129787672942031250 [04/13/12 05:08:14.203]
Access time [utc]:  129787672942031250 [04/13/12 05:08:14.203]
MFT Changed time [utc]:  129936087500625000 [10/01/12 23:45:50.062]
Create time [utc]:  129787672941093750 [04/13/12 05:08:14.109]
Size allocated:  0x00005000 [20480]
Size used:      0x00004d31 [19761]
Type entry:     0x00002020 [ARCHIVE, CONTENT_NOT_INDEXED]
Source type:    indx alloc
```

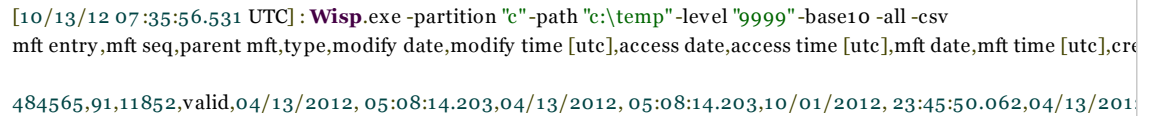
WISP (Windows INDX Slack Parser) also from TzWorks is another potential directory lister. By design it parses **INDX** records found in different \$metadata than the \$MFT. Note that Microsoft apparently does not acknowledge (<http://www.google.com/search?&rls=en&q=site:microsoft.com+ntfs+indx+records>) this "INDX" terminology, instead preferring \$I30 or \$INDEX_ROOT and \$INDEX_ALLOCATION. I don't know enough to say whether the data is exclusive to \$I30 or whether it is linked in from \$MFT or whether this utility access both locations to obtain the output. Regardless, the data produced is just as useful, in fact perfectly adequate if only it was formatted correctly! Unlike most earlier examples, an output header is not present. **Shortcomings:** The common theme of MM/DD/YYYY and UTC and others remain. But needless to say, this particular command is ill-suited for a directory listing without substantial post-processing since it used 14 lines output per file!

NOTE-1: WISP prints all entries denoted as either "valid" or "slack" representing actual files or unused space respectively, and the utility offers the choice of corresponding switches **-valid** or **-slack** or **-all**. Also note that like NTFSDIR it makes subdirectory recursion difficult ***and*** it changes the switch name from **-recurse** to **-level**, and it specifically tells you that "recurse # levels [default is 0 levels]" which instead of meaning infinite, it means only the specified folder. So

the 9999 kludge is also used here.

NOTE-2: From the displayed output you can see the all required fields are there, and then some. Note that in these examples I jumped immediately to using the switches **-base10** and **-all**, rather than reproducing much of what we already have seen previously.

EXAMPLE 13 ... Wisp.exe -partition c -path c:\temp -level 9999 -base10 -all -csv >Output.txt



```
[10/13/12 07:35:56.531 UTC] : Wisp.exe -partition "c" -path "c:\temp" -level "9999" -base10 -all -csv
mft entry,mft seq,parent mft,type,modify date,modify time [utc],access date,access time [utc],mft date,mft time [utc],cre
484565,91,11852,valid,04/13/2012, 05:08:14.203,04/13/2012, 05:08:14.203,10/01/2012, 23:45:50.062,04/13/2012, 05:08:14.203
```

The same command as example 12 but with the added switch **-csv**. The data is now contracted onto one line and with much post-processing it could be useful as a file list. Like earlier examples, the output header strips off the **>Output.txt** from the displayed command, but it is definitely there. Additionally, it embedded those quotes you see, but they were not in the batch file command. The actual executed command used in a batch file is listed after the example number. **Shortcomings:** The common theme of MM/DD/YYYY and UTC and others remain. The inconsistent delimiter problem is present, mostly using **","** (comma) but sometimes (in the time fields) using **" "** (comma,space).

Conclusion

As you can see, there are many kinds of outputs just from these couple of utilities. The most frustrating problem is that these particular utilities use UTC time with no option to use local (i.e., what you would see when accessing the disk in a file manager or viewing standard DIR listings). So what I am wondering is if any others have looked into this and tested various tools and if possible, please share your suggestions and recommendations. More examples will be posted above as I test other utilities.

Presently I am aware of two other possibilities which apparently also make use of the \$MFT. One is found in the Gnuwin32 package called **Ntfsprogs** which contains **Ntfsls.exe** ("list directory contents on an NTFS filesystem"). The other possibility is called **Mft2csv** and is still in recent development authored by Joakim Schicht (who if I'm not mistaken is a member here and also at reboot.pro aka boot-land.net). The documentation link seems to be broken for Mft2csv, and the NtfsProgs package has no examples I can find. So I haven't yet tried either but will shortly once I find some example commands. **Other suggestions are most welcome!**

The above mentioned *Forensic Tools Prototypes* by TzWorks are from a collection of 20 available at their website (linked below) and come as both 32 and 64-bit Windows binaries. Please don't let the mention of those few quirks or bugs color your opinion of them. There are some **very** impressive tools there. In particular the one called **JP** nicely parses the NTFS journal into a text log, **CAFAE** pulls common and unique artifacts from the registry, **YARU** has a GUI to view nearly impossible to access hives, and there are many more. Perhaps a thread detailing these promising tools would be an idea for later.

Related Information ...

- **Forensic Tools Prototypes** (<http://www.tzworks.net/prototypes.php>) (TzWorks.net)
- **Reading MFT** (<http://social.msdn.microsoft.com/Forums/en-US/csharpgeneral/thread/c1550294-d121-4511-ac32-31551497f64e>) (Social.Msdn.Microsoft)
- **MFT-Access - Reading & parsing the Master File Table on NTFS Filesystems** (<http://www.autoitscript.com/forum/topic/94269-mft-access-reading-parsing-the-master-file-table-on-ntfs-filesystems/>) (AutoItScript.com)
- **mft2csv** (<http://code.google.com/p/mft2csv/wiki/mft2csv>) - Joakim Schicht (Code.Google.com)
- **NtfsProgs for Windows** (<http://gnuwin32.sourceforge.net/packages/ntfsprogs.htm>) (GnuWin32)
- **NTFS** (<http://en.wikipedia.org/wiki/NTFS>) (Wikipedia)
- **INDXParse** (<http://www.williballenthin.com/forensics/indx/index.html>) (Willi Ballenthin)

Originally Posted on 2012-10-17. Please report any typos or mistakes!

2012-10-17 ... EDIT: clarity, typos

2012-10-18 ... EDIT: TzWorks link was fixed. There is a strange bug in the forum (or maybe Opera?) when doing copy/paste of text in "Full Editor" causing random artifacts being inserted into the middle of words!

Edited by CharlotteTheHarlot, 18 October 2012 - 08:38 AM.

- App Management
- Policy Management

- Patch Management
- Software Deployment

Download 



[How to remove advertisement from MSFN](#)

CharlotteTheHarlot

Posted 17 October 2012 - 02:52 AM

[reserved]

CharlotteTheHarlot

Posted 17 October 2012 - 02:52 AM

[reserved]

jaclaz

Posted 17 October 2012 - 06:10 AM

'CharlotteTheHarlot', on 17 Oct 2012 - 04:52 AM, said:

Presently I am aware of two other possibilities which apparently also make use of the \$MFT. One is Gnuwin32 package called **Ntfsprogs** which contains **Ntfs.exe** ("list directory contents on an NTFS filesystem"). The other possibility is called **Mft2csv** and is still in recent development authored by Joakim Schicht (who if I'm not mistaken is a member here and also at reboot.pro aka boot-land.net). The documentation link seems to be broken for Mft2csv, and the NtfsProgs package has no examples I can find. So I haven't yet tried either but will shortly once I find some example commands. **Other suggestions are most welcome!**

What is the problem with mft2csv?

JFYI, the "development" thread is here (and yes, **joakim** is a member there besides MSFN and reboot.pro):

<http://www.forensicsf...er=asc/start=0/>

(<http://www.forensicsfocus.com/Forums/viewtopic/t=8010/postdays=0/postorder=asc/start=0/>)

but AFAICR mft2csv is a \$MFT parser, you need to provide it with a \$MFT (and that's where NTFS_File_Extractor may come into play):

<http://code.google.c.../downloads/list> (<http://code.google.com/p/mft2csv/downloads/list>)

Both are written with autoit and the source is available, but right now they should be GUI only.

The actual output of mft2csv (once you "feed it" with a \$MFT) contains more info that you will ever want to know, or if you prefer, your "requirements" are a "small subset" of the capabilities of that tool.

Possibly a good idea would be to contact the Authors of tools like ndff or tfind:

<http://ndff.hotbox.ru/en/index.html> (<http://ndff.hotbox.ru/en/index.html>)

<http://www.deadnode.org/sw/tfind/> (<http://www.deadnode.org/sw/tfind/>)

or other similar tools, that already have the "main" \$MFT parsing engine. 🙄

jaclaz

CharlotteTheHarlot

Posted 17 October 2012 - 07:09 AM

'jaclaz', on 17 Oct 2012 - 08:10 AM, said:

What is the problem with mft2csv?

JFYI, the "development" thread is here (and yes, **joakim** is a member there besides MSFN and reboot.pro):

<http://www.forensicsf...er=asc/start=0/>

(<http://www.forensicsfocus.com/Forums/viewtopic/t=8010/postdays=0/postorder=asc/start=0/>)

but AFAICR mft2csv is a \$MFT parser, you need to provide it with a \$MFT (and that's where NTFS_File_Extractor may come into play):

<http://code.google.com/p/mft2csv/downloads/list> (<http://code.google.com/p/mft2csv/downloads/list>)

Both are written with autoit and the source is available, but right now they should be GUI only.

The actual output of mft2csv (once you "feed it" with a \$MFT) contains more info that you will ever want to know, or if you prefer, your "requirements" are a "small subset" of the capabilities of that tool.

Possibly a good idea would be to contact the Authors of tools like ndff or tfind:

<http://ndff.hotbox.ru/en/index.html> (<http://ndff.hotbox.ru/en/index.html>)

<http://www.deadnode.org/sw/tfind/> (<http://www.deadnode.org/sw/tfind/>)

or other similar tools, that already have the "main" \$MFT parsing engine. 🙄

Thanks Jaclaz. I was hoping you would throw out some ideas. I have no problem grabbing the metadata off disks, feeding them into programs, etc. I'm just trying to fine-tune the output listing to that narrow set of fields.

RE: **Mft2csv**, I didn't mean to imply of any problem whatsoever, all I said was I hadn't tested it yet. There is no documentation I could find (switches, etc) and I perused the **.au3** source and nothing was obvious. So I just wanted to be absolutely sure I had the instructions before testing and reporting on it. It's just the right thing to do. So I think you are saying **Mft2csv** is GUI not CLI, correct? If so, no switches to worry about!

I will look into those others you mention as well. Please let me know if you come across any others (DIR file list alternatives).

P.S. Actually a while back I googled for references to TzWorks and after eliminating links to their own website, amazingly there seemed to only be about 5 people in the world even aware of those tools. Not surprisingly, one of those people was **You** (<http://www.911cd.net/forums//index.php?showtopic=24955&st=0&p=17directory1833&#entry171833>)! 🙄👍

jaclaz

Posted 17 October 2012 - 09:20 AM

Yep, the "usual" approach (since the *need* for details on files is more "felt" in forensics and to a much lesser extent in data recovery) is to extract/make a copy of the \$MFT and then parse it.

If you search for "\$MFT parsing" or "\$MFT parser" or "NTFS parser" you will find a few (usually python) scripts, such as:

<http://www.integriography.com/> (<http://www.integriography.com/>)

As well, if you are into putting something together yourself there are a few libraries around:

<http://www.codeproject.com/Articles/74128/NTFS-MFT-deleted-files>

<http://www.codeproject.com/Articles/81456/An-NTFS-Parser-Lib>

and some AHK tool/code that can probably be "adapted" 🙄:

<http://www.autohotkey.com/community/viewtopic.php?f=13&t=85072> (<http://www.autohotkey.com/community/viewtopic.php?f=13&t=85072>)

jaclaz

joakim

Posted 17 October 2012 - 02:39 PM

Just adding some input regarding my own stuff. mft2csv does in fact let you choose (when starting the program) how timestamps are given (local time vs utc). Although not implemented yet, I had a wish to implement configurable output. The format of timestamps can also be changed without too much fuzz. The source is provided and I actually asked for suggestions at forensicfocus regarding timestamp format.

The drawback with my tool(s) as they are right now, is:

- Slower processing than most other.
- Missing full path

However implementing full path is not that hard, so we'll see..

Not sure about the current state of analyzeMFT, but I was in contact with the author some time ago regarding missing fixups

(which is important). That tool was of very much help for me in the beginning, and I studied his script well back then.

CharlotteTheHarlot

Posted 18 October 2012 - 05:50 AM

'joakim', on 17 Oct 2012 - 4:39 PM, said:

Just adding some input regarding my own stuff. mft2csv does in fact let you choose (when starting the program) how timestamps are given (local time vs utc). Although not implemented yet, I had a wish to implement configurable output. The format of timestamps can also be changed without too much fuzz. The source is provided and I actually asked for suggestions at forensicfocus regarding timestamp format.

The drawback with my tool(s) as they are right now, is:

- Slower processing than most other.
- Missing full path

However implementing full path is not that hard, so we'll see..

Not sure about the current state of analyzeMFT, but I was in contact with the author some time ago regarding missing fixups (which is important). That tool was of very much help for me in the beginning, and I studied his script well back then.

Hi Joakim, Thanks for stopping in! You've got some very useful projects underway.

Regarding Mft2csv, as I say I didn't run it yet, is it GUI or CLI? Also, Is the speed a consequence of AutoIt overhead? Is there documentation and can other modify the source (with attribution)?

What I am aiming for is a replacement DIR that uses the \$MFT or other metadata, for both completeness sake (all existing files) and for speed. Preferably not CSV but plain text like DIR, or the best case scenario with an option for either. Is there any chance of someone adapting Mft2csv into these requirements? If you can think of any other candidates please let me know. 🙏🙏🙏

P.S. has anyone tested the GnuWin32 NTFSPROGS files?

jaclaz

Posted 18 October 2012 - 07:29 AM

Other candidates (again NOT exactly doing what required):

<http://home.comcast...fsfastfind.html> (<http://home.comcast.net/~lang.dennis/console/ntfsfastfind/ntfsfastfind.html>)
<http://sourceforge.n...ts/ntfs-search/> (<http://sourceforge.net/projects/ntfs-search/>)

This one might (or possibly "will"):

<http://malware-hunte...-forensic-tool/> (<http://malware-hunters.net/2011/03/30/introducing-mftdump-forensic-tool/>)
<http://malware-hunte...category/tools/> (<http://malware-hunters.net/category/tools/>)

Some .NET ☺ thingy:

<http://sourceforge.n...cts/ntfsreader/> (<http://sourceforge.net/projects/ntfsreader/>)

jaclaz

Edited by jaclaz, 18 October 2012 - 07:31 AM.

joakim

Posted 18 October 2012 - 02:00 PM

The speed is partly due to autoit scripting overhead, but also coding style I presume. Source is available in download section along with compiled binaries, and in the source section where you can browse it and track changes. You can modify and use it to whatever you like. I don't have time to support it right now, and likely not this year either. Point is it takes a lot of time..

Btw, it is gui compiled in current version.

xpclientPosted 19 October 2012 - 01:45 AM

Btw, doesn't the Everything search product from Voidtools work by directly indexing the NTFS MFT?

CharlotteTheHarlotPosted 19 October 2012 - 05:00 AM

'xpclient', on 19 Oct 2012 - 03:45 AM, said:

Btw, doesn't the Everything search product from Voidtools work by directly indexing the NTFS MFT?

Could be. I have never used it myself probably because I saw "indexing" in the description and I prefer brute force searches. It would probably have to be a hybrid though because if it indexes content, it would necessarily need more than just the \$MFT and require accessing data sectors also (well, unless it was a very small file completely held within the \$MFT). But I do believe the common comments that they optimized their engine faster than the indexers found in XP or Vista or 7 or whatever because they can be a nightmare. I always kill them first thing. 😊

Here is the link: [VoidTools Everything Search Engine \(http://www.voidtools.com/\)](http://www.voidtools.com/) and the [FAQ \(http://www.voidtools.com/faq.php\)](http://www.voidtools.com/faq.php) and [Downloads \(http://www.voidtools.com/download.php\)](http://www.voidtools.com/download.php).

xpclientPosted 19 October 2012 - 05:05 AM

I use it daily. It reads the MFT and stores the names in a database of less than 10 MB. Obviously file names only-"indexing", no file contents. So "indexing" takes less than 30 seconds for even drives as large as 2-3 TB and you can get a list of the entire HDD's file system and can search the database! It's pretty amazing.

CharlotteTheHarlotPosted 19 October 2012 - 06:11 AM

'xpclient', on 19 Oct 2012 - 07:05 AM, said:

I use it daily. It reads the MFT and stores the names in a database of less than 10 MB. Obviously file names only-"indexing", no file contents. So "indexing" takes less than 30 seconds for even drives as large as 2-3 TB and you can get a list of the entire HDD's file system and can search the database! It's pretty amazing.

Ahh, name-only indexing. Cool. That would explain the \$MFT usage (and the speed!). When you need content searching just do good old brute force.

Have you ever tried to use it on a standard account? I also wonder if the indexer or the search function would return the name of a test file with "deny" on everything.

xpclientPosted 19 October 2012 - 07:20 AM

Everything requires administrator permissions. I don't understand what you mean by last line.

jaclazPosted 19 October 2012 - 11:42 AM

'xpclient', on 19 Oct 2012 - 03:45 AM, said:

Btw, doesn't the Everything search product from Voidtools work by directly indexing the NTFS MFT?

Yes it does, and it is fastish 🙄, but as you already said, it does it by indexing the files and it does not provide the level of details "required" by OP.

Additionally, AFAICR, the "command line" part of it is just a "remote command" for the index (thus, to have "real-time" data, you need to have the GUI running):

```
C:\Everything>es -h
```

-r **Search** the database **using** a basic POSIX regular expression.
 -i **Does** a **case** sensitive search.
 -w **Does** a whole word search.
 -p **Does** a full path search.
 -h --help **Display this** help.
 -n **<num>** **Limit** the amount of results shown to **<num>**.
 -s **Sort by** full path.

For the record it does exist an "enhanced" version of the es program (rigorously UNtested by me) here:

<http://forum.voidtools.com/viewtopic.php?f=5&t=322> (<http://forum.voidtools.com/viewtopic.php?f=5&t=322>)
<https://sites.google.com/site/xixinxing/programs/nothingforeverything> (<https://sites.google.com/site/xixinxing/programs/nothingforeverything>)

jaclaz

CharlotteTheHarlot

Posted 20 October 2012 - 03:41 AM

'xpclient', on 19 Oct 2012 - 09:20 AM, said:

Everything requires administrator permissions. I don't understand what you mean by last line.

I meant what if you (as Admin) changed security on a test file or test folder to "deny" for standard users for READ, LIST, TRAVERSE, etc (special permissions), and then logged in as standard user and tried indexing and searching those files/folders.

Not important, but just curious if Everything Search bypassed ACLs. One of the big problems with standard FIND programs (including Windows Search) is getting different results depending on what account it is running under.

Although, having the indexer and search engine GUI run elevated might alleviate it to some degree, or perhaps it already runs elevated. I don't know, just spinning my wheels since I never ran that program.

xpclient

Posted 20 October 2012 - 03:55 AM

Ah, now I see what you mean. Everything elevates itself, always runs as admin. Even if you set deny permissions to entire Administrators group, Everything finds it. Make sure you use the latest EXE released in the forums: [1.2.1.451a](http://www.voidtools.com/Everything-1.2.1.451a.zip) (<http://www.voidtools.com/Everything-1.2.1.451a.zip>) What it can't do is index contents. It can't index network paths either. Its developer was last seen active in their forums in 2011 but forum registrations are disabled so no way to contact him. There is a cmd line as jaclaz mentioned, and also an SDK but there isn't a 64-bit one. 😞

Edited by xpclient, 20 October 2012 - 04:01 AM.

[Back to Windows XP](#)



[How to remove advertisement from MSFN](#)

MSFN Forum → Microsoft Software Products → Windows XP