



opensourceforensics

Open Source Forensic Analysis

[Project Home](#)[Downloads](#)[Wiki](#)[Issues](#)[Source](#)

Search

Current pages



for

Search

analyzeMFT

*analyzeMFT - a Python tool to deconstruct the Windows NTFS \$MFT file*Updated Nov 11, 2011 by [dko...@gmail.com](#)

- [Introduction](#)
- [Version](#)
- [Background](#)
- [Installation and use](#)
- [Output](#)
- [Sample Output](#)
- [Future work](#)
- [Change log](#)

Introduction

analyzeMFT.py is designed to fully parse the MFT file from an NTFS filesystem and present the results as accurately as possible in a format that allows further analysis with other tools. At present, it parses the attributes from a \$MFT file to produce the following output:

- Record Number
- Good - if the entry is valid
- Active - if the entry is active
- Record type - the type of record
- Record Sequence - the sequence number for the record
- Parent Folder Record Number
- Parent Folder Sequence Number
- For the standard information attribute:
 - Creation date
 - Modification date
 - Access date
 - Entry date
- For up to four file name records:
 - File name
 - Creation date
 - Modification date
 - Access date
 - Entry date
 - Object ID
 - Birth Volume ID
 - Birth Object ID
 - Birth Domain ID
- And flags to show if each of the following attributes is present:
 - Standard Information, Attribute List, Filename, Object ID, Volume Name, Volume Info, Data, Index Root, Index Allocation, Bitmap, Reparse Point, EA Information, EA, Property Set, Logged Utility Stream
- Notes/Log - Field used to log any significant events or observations relating to this record
 - std-fn-shift - Populated if anomaly detection is turned on. Y/N. Y indicates that the FN create date is later than the STD create date.
 - usec-zero - Populated if anomaly detection is turned on. Y/N. Y indicates that the STD create date's microsecond value is zero.

For each entry in the MFT a record is written to an output file in CSV format.

analyzeMFT will run on any system with Python installed. A standalone Windows executable is also available,

Contributions and suggestions for improvement are quite welcome.

analyzeMFT is Copyright (c) 2010 David Kovar. All rights reserved. This software is distributed under the Common Public License 1.0.

Major contributions from Matt Sabourin.

Version

Currently at version 2.0. The 2.0 version has some additional features, and additional limitations. Consider using the 1.7 version if you're working with large \$MFT files as you might run out of memory using the 2.0 version.

Background

My original inspiration was a combination of MFT Ripper (thus the current output format) and the SANS 508.1 study guide. I couldn't bear to read about NTFS structures again, particularly since the information didn't "stick". I also wanted to learn Python so I figured that using it to tear apart the MFT file was a reasonably sized project.

Many of the variable names are taken directly from Brian Carrier's The Sleuth Kit. His code, plus his book "File System Forensic Analysis", was very helpful in my efforts to write this code.

The output format is almost identical to Mark Menz's MFT Ripper. His tool really inspired me to learn more about the structure of the MFT and to learn what additional information I could glean from the data.

I also am getting much more interested in timeline analysis and figured that really understanding the the MFT and having a tool that could parse it might serve as a good foundation for further research in that area.

Installation and use

Usage: analyzeMFT.py [options]

Options:

```
--version          show program's version number and exit
-h, --help         show this help message and exit
-f FILENAME, --filename=FILENAME
                   [Required] Name of the MFT file to process.
-d, --debug        [Optional] Turn on debugging output.
-p, --fullpath     [Optional] Print full paths in output (see comments
                   in code).
-n, --fntimes      [Optional] Use MAC times from FN attribute instead of
                   SI attribute.
-a, --anomaly      [Optional] Turn on anomaly detection.
-b BODYFILE, --bodyfile=BODYFILE
                   [Optional] Write MAC information in mactimes format
                   to this file.
-m MOUNTPPOINT, --mountpoint=MOUNTPPOINT
                   [Optional] The mountpoint of the filesystem that held
                   this MFT.
-g, --gui          [Optional] Use GUI for file selection.
-o OUTPUT, --output=OUTPUT
                   [Optional] Write analyzeMFT results to this file.
```

Source: The source code can be downloaded from Source page of this site . You will need Python installed on your system to run it. I recommend ActivePython, free, from [here](#).

You need to have the tkinter python module and may need tk/tcl as well. The first thing to try is installing the tkinter module for your platform. More details can be found [here](#). If you don't want to install tk/tcl, just set the "noGUI" flag to 'True' in the code and it will not try to import the modules.

Binary executable: The binary version is often out of date. Use the source (Luke) whenever possible.

The installer for a standalone binary is available [here](#). A couple of notes: It often lags (way)behind the source version. If at all possible, use the source. It installs the Microsoft Visual C++ 2008 Redistributable Package. This provides the runtime libraries required. If you want to install this for yourself, it can be found [here](#). On Windows XP, the Startup Menu install works correctly and if you start the application in this manner, the GUI comes up. On Windows 7, the startup parameter doesn't get set by the installer so the application starts up and immediately exits. You can run it from the command line without problems.

Output

The output is currently written in CSV format. Due to the fact that Excel automatically determines the type of data in a column, it is recommended that you write the output to a file without the .csv extension, open it in Excel, and set all the columns to "Text" rather than

"General" when the import wizard starts. Failure to do so will result in Excel formatting the columns in a way that misrepresents the data.

I could pad the data in such a way that forces Excel to set the column type correctly but this might break other tools.

Sample Output

Sample output:

```
"Record Number","Good","Active","Record type","Parent Folder","Record Sequence","Filename #1","Std Info Creation date"
"0","Good","Active","File","5 - 5","1","$MFT","2007/07/31 19:16:13.734373","2007/07/31 19:16:13.734373","2007/07/31 19:16:13.734373"
"110575","Good","Inactive","0","5422 - 5426","3","TRANSFERMGR.EXE-24D2A23F.pf","2009/12/27 18:35:57.625000","2009/12/27 18:35:57.625000"
```

Future work

Future work:

1. Figure out how to write the CSV file in a manner that forces Excel to interpret the date/time

fields as text. If you add the .csv extension Excel will open the file without invoking the import wizard and the date fields are treated as "General" and the date is chopped leaving just the time.

1. Add version switch
2. Add "mfr" switch - produce MFT Ripper compatible output
3. Add "extract" switch - extract or work on live MFT file
4. Finish parsing all possible attributes
5. Look into doing more timeline analysis with the information
6. Improve the documentation so I can use the structures as a reference and reuse the code more effectively
7. Clean up the code and, in particular, follow standard naming conventions
8. There are two MFT entry flags that appear that I can't determine the significance of. These appear in the output as Unknown1 and Unknown2
9. Parse filename based on 'nspace' value in FN structure
10. Test it and ensure that it works on all major Windows OS versions
11. Output HTML as well as CSV

See other ToDos in the code

Change log

Updates:

(Most recent on top starting with 2.0)

Version 2.0: Matt Sabourin - Created an object-oriented version of analyzeMFT.py. Most of the MFT analysis code and other logic was retained from the original version (along with the comments). The OO version is structured for importing the module directly into the python interpreter to allow for manual interaction with the MFT. The module can also be imported into other python scripts that need to work with an MFT. In addition to switching to OO, I added following code:

- Allow printing of full path file names
- Allow use of time values from FN attribute instead of SI
- Fixed issue with bodyfile output and newest version of mactimes - use longs vs floats for times
- Bodyfile now includes MFT (physical) record number
- Bodyfile now includes zero for MD5 value
- anomalyDetect also flags SI attributes with zero microsecond

Fixed (?) potential issues with analyzeMFT output

- If a record w/o an SI attribute follows a record that has an SI attribute, the second record was printing times for the previous record's SI attribute
- Printing out name, times for FN0 (first FN) attribute

Version 1.x notes

Version 1.0:

- Initial release

Version 1.1:

- Split parent folder reference and sequence into two fields. I'm still trying to figure out the significance of the parent folder sequence number, but I'm convinced that what some documentation refers to as the parent folder record number is really two values - the parent folder record number and the parent folder sequence number.

Version 1.2:

- Fixed problem with non-printable characters in filenames. Any Unicode character is legal in a filename, including newlines. This presented some problems in my output. Characters that do not render well are now converted to hex and a note is added to the Notes column indicating this. (I've learned a lot about Unicode since I first wrote this.)
- Added "compile time" flag to turn off the inclusion of any GUI related modules and libraries for systems missing tk/tcl support. (Set noGUI to True in the code)

Version 1.3:

- Added new column to hold log entries relating to each record. For example, a note stating that some characters in the filename were converted to hex as they could not be printed.

Version 1.4:

- Credit: Spencer Lynch. I was misusing the flags field in the MFT header. The first bit is Active/Inactive. The second bit is File/Folder.

Version 1.5:

- Fixed date/time reporting. I wasn't reporting useconds at all.
- Added anomaly detection. Adds two columns:
 - std-fn-shift: If Y, entry's FN create time is after the STD create time
 - usec-zero: If Y, entry's STD create time's usec value is zero

Version 1.6:

- Various bug fixes

Version 1.7:

- Bodyfile support, with many thanks to Dave Hull

► [Sign in](#) to add a comment

©2011 Google - [Terms](#) - [Privacy](#) - [Project Hosting Help](#)

Powered by [Google Project Hosting](#)