HACKING EXPOSED
COMPUTER FORENSICS BLOG
BY DAVID COWEN

Monday, January 7, 2013

NTFS Triforce - A deeper look inside the artifacts

Hello Reader,

In our last post we discussed at a high level the relationship between the $MFT, $LOGFILE and $USNJRNL. In this post we will go into detail of the structures we can recover from each of the three and how they link allowing us to determine the historical changes made to a file or directory.

$MFT - The Master File Table is a pretty well understood artifact. MFT structures are fully documented and there are a variety of tools out there for parsing it. With that said, I'm not going into any depth on how the MFT works but instead just highlight the two structures we are interested in.

(Thanks to Mike Wilkinson for making these MFT data structure diagrams I am referencing below. You can find the full version of his NTFS cheat sheet here **http://www.writeblocked.org/resources/ntfs_cheat_sheets.pdf**)

The first is the File Record shown below:

File Record Segment Header

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	F	I	L	E	Update Seq array offset		Update Seq array size		$LogFile Sequence Number							
1	Seq no		Hard Link Count		1st attrib offset		Flags		Used size of file record				Allocated size of file record			
2	File reference to base file record								Next attrib ID				MFT Record No			
3	default location of update seq array (size determined by seq size)						Reserved for update sequence array?									
	Reserved for sequence array?								Common location of 1st attrib							

When a file is created, modified, or deleted this is the structure that gets added, changed, or updated. The field in the upper right at offset 0x08 labeled $Logfile Sequence Number or LSN is how the MFT refers to the most recent change recorded in the $logfile. Each $logfile record has an associated LSN, however the LSN is updated in the file record to correspond to the most recent change. There is no record that I'm aware of that shows what LSNs a file record previously had. The MFT Record Number is a unique identifier for this file record, and if we have a way to link a change to it then it becomes easy to associate historical changes we recover to indicate which MFT file record they are referencing.

The $USNJrnl keeps the MFT Record Number to indicate which file it is operating on and the Parent Record number to reflect what directory that MFT file record resided in. If a $logfile entry records a change then that change can be easily linked back to the MFT file record number's LSN if it's the last change made to that file record.

The file record however is not the only record/attribute we care about in the MFT for our triforce historical analysis powers, we also care a lot about the Standard Information record shown below:

$Standard_Information

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Date Created*								Date Modified							
10	Date MFT record modified								Date Accessed							
20	Flags				Max Versions				Version Num				Class ID			
30	Owner ID				Security ID				Quota Charged							
40	Update Sequence Number															

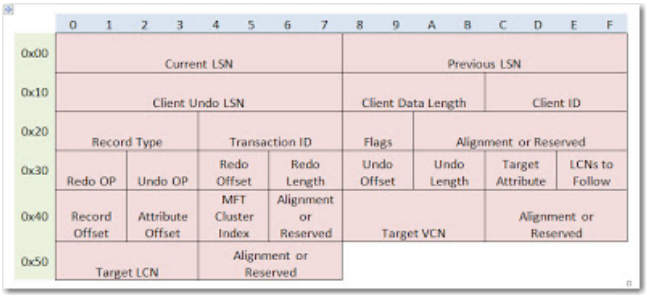
*Time values are in 100 nanoseconds since January 1, 1601 UTC

If a time stamp, owner id or SID of a file changes then it's the standard information block/attribute that gets written to the $logfile and not the entire file

record with all its attributes. This was a problem before we found the triforce linkage because as you can see the standard information block does not refer to the file record number. We had to determine which MFT entry a $logfile record was pointing to by either the LSN (which is captured in the Logfile header per recorded change) and hope it hasn't been updated again. Alternatively we could determine the location of the MFT entry by doing some math using the VCN (virtual cluster number) and the MFT cluster ID recorded in the $logfile. Relying on the physical location in the MFT is also problematic because a defrag can remove deleted entries and change the VCN where the entry resides leading to false positives of which $logfile record points to which MFT record.

The good news here is that as you can see at offset 0x40 the standard information attribute does record the update sequence number! The update sequence number in turn will point to the file record number and parent file record number as discussed above. This means that through the link between the $USNJrnl and the $MFT we can associate a change made to the standard information attribute from the $logfile to the $USNJrnl which links back to a specific $MFT file record number. This is a reliable identifier as the file record number's value does not change based on system activity! This then leads us to the $logfile structures.

$Logfile - Every change recorded in the $logfile starts with a header as shown below:



The LSN here relates back to the file record entry inside the MFT for the change that is being recorded. The LSN for a file record in the MFT will be updated to reflect the most current $logfile entry for that file. Meaning the LSN for a file will change with every change recorded. That means than any $logfile entry whose recorded change does not reference either the USN or the MFT record number can only have its corresponding MFT record determined by doing a calculation using the recorded VCN seen at offset 0x48 above.

Why does the $logfile record the VCN? The process of repairing the file system using the $logfile is to overlay the data stored in the $logfile over the areas where a transaction failed to complete successfully. This allows the file system to be rolled back (using the undo records) or have a change reapplied (using the redo records) by just overwriting what previously located at those VCNs.

What comes after the LSN record header will vary on what change took place, the $logfile is storing the raw MFT record/attribute that has been modified so any MFT entry could exist in the $logfile. We focus on the File records and the Standard information attribute records as they reveal the most about changes occurring to a file. There are other MFT records/attributes that could be of interest to you and they also exist in the $logfile. Any change made to a MFT record/attribute will be recorded in the $logfile, the hard part is then referencing that logged change to the actual MFT record being modified to know which file record it relates to. So you can imagine that after every LSN header you have a copy of the MFT record/attribute being changed reflecting its before (undo) and after (redo) states.

Since there are no other $logfile structures other than the LSN header, RCRD header and Restart areas we are reliant on what is being recorded by the MFT record being changed to exactly know which file is being modified. When we are lucky (like we are with file records and $standard_information records) we get a link back to a unique file reference number. When we are unlucky (resident data found in $DATA attribute records) we have to rely on some math using the VCN and MFT Cluster index stored in the LSN header to determine what location within the MFT the record is pointing to. It's this possibility for false positives that

- ► December (30)
- ► November (33)
- ► October (31)
- ► September (29)
- ► August (33)
- ► July (32)
- ► June (8)
- ► May (1)
- ► April (2)
- ► March (2)
- ▼ January (2)
  - NTFS Triforce - A deeper look inside the artifacts...
  - Happy new year, new post The NTFS Forensic Triforc...
- ► 2012 (13)
- ► 2011 (11)
- ► 2010 (1)
- ► 2009 (9)
- ► 2008 (2)

Categories

- accessdata (4)
- active directory (1)
- admissibility (2)
- aduc (1)
- aim (2)
- aix (1)
- ajax (1)
- anjp (4)
- answer key (1)
- april fools (1)
- arsenal (2)
- attachments (3)
- austin (1)
- awards (1)
- back to basics (1)
- best finds (2)
- beta (2)
- bitlocker (1)
- blackberry enterprise server (1)
- book (2)
- bsides (3)
- carved (1)
- ccdc (1)
- cd burning (2)
- ceic (2)
- computer forensics (9)
- conference schedule (1)
- contest (80)
- court approved (1)
- credentials (1)
- Daily Blog (2)
- dbir (2)
- deviceclasses (1)
- dfir summit (2)
- dingo stole my baby (1)
- dirty file system (1)
- dvd burning (1)
- email recovery (1)
- emdmgmt (1)
- Encyclopedia Forensica (1)
- evidence (1)
- external drives (2)
- fde (1)
- for408 (1)
- for668 (2)
- forensic 4cast (1)
- forensic lunch (34)

keeps these records out of the public version of our $logfile parser.

Note: We will go into even more detail of the $logfile structures when we do the big $logfile post which is coming with the tool release I promise.

$USNJrnl - The USNJrnl or Update Sequence Number Journal has a pretty simple structure compared to the rest we've talked about and is fully documented as shown below:

```
typedef struct {
  DWORD         RecordLength;
  WORD          MajorVersion;
  WORD          MinorVersion;
  DWORDLONG     FileReferenceNumber;
  DWORDLONG     ParentFileReferenceNumber;
  USN           Usn;
  LARGE_INTEGER TimeStamp;
  DWORD         Reason;
  DWORD         SourceInfo;
  DWORD         SecurityId;
  DWORD         FileAttributes;
  WORD          FileNameLength;
  WORD          FileNameOffset;
  WCHAR         FileName[1];
} USN_RECORD_V2, *PUSN_RECORD_V2, USN_RECORD, *PUSN_RECORD;
```

Sorry no fancy hex offset data structure for this yet, just the record structure as taken from **Microsoft's documentation of the USNJrnl**. As you can see for purposes of linking back $standard_information structures stored in the $logfile to the MFT we have the matching USN stored here as the sixth element down. Since each USNJrnl entry, and thus each open/close of a file, has a unique USN assigned we have a great lasting artifact to look for when trying to match $standard_information records back to MFT records. The fourth and fifth items in the record entry link back to the MFT for not only the file record number but also the directory the file was located in as seen in the parent record number.

Taken just on its own the USNJrnl is a fantastic source of historical information that more examiners are beginning to utilize, you can get even more information out if it by taking it a step further. If you were to mine out all the unique USN records into a database table you could group them by file reference number to see all the changes including the renaming of a file or its movement between directories.This is because the MFT file record number (shown in the MSDN screenshot above as a reference number) does not change no matter how many times the file record or attributes change. Renaming a file, moving a file, editing its time stamps, filling it with random data, etc... none of these actions will change the file record number. What utilizing the triforce gets us is more granular details of those attribute changes that only exist in the $logfile extrapolated out through the $USNJrnl to a MFT file record.

Putting it all together - So that was a lot of words up there, if you read the last post you got the same information at a very high level but now you can see at a much deeper level how these things sync up. I don't believe that the developers actually intended this relationship to exist, or else I would expect more syncing for more record types stored in the $logfile, we just again get a happy overlap between what a developer made and what analysis can reveal to us.

If you followed everything I wrote above you will see that using the power of the NTFS triforce we can recover and identify:
1. The change of ownership of a file ($logfile)
2. The change of a file's SID (if that were to happen) ($logfile)
3. The changing of timestamps ($logfile)
4. The movement of files between directories ($logfile and $USNJrnl)
5. The renaming of files (common during wiping) ($logfile and $USNJrnl)
6. The summary of actions taken against a file ($USNJrnl)
7. The changing of attributes to a file, important for things like tracking hard links to determine CD Burns ($logfile)

We can do all of these with little chance of error thanks to the combination of these three data sources. Additionally we can recover granular historical changes to files. Depending on your location in the DFIR spectrum (from digital forensics analyst, incident responder to malware analyst or all of the above) you will have different uses for this information. We are very excited about thee triforce and we are extending our $logfile parser to include these sources, of

- **forensic tips** (1)
- **fraud** (3)
- **ftk** (2)
- **ftk 2** (1)
- **full disk encryption** (1)
- **gcfe** (1)
- **go bag** (7)
- **howto** (7)
- **IE10** (1)
- **imaging** (1)
- **indepth** (1)
- **information theft** (1)
- **infosec pro guide** (1)
- **intern** (1)
- **Interview** (1)
- **ir** (2)
- **json** (1)
- **jump lists** (1)
- **last logon** (1)
- **libvshadow** (1)
- **linux** (6)
- **linux-3g** (4)
- **lnk files** (3)
- **log analysis** (4)
- **log2timeline** (1)
- **love notes** (1)
- **macmini** (1)
- **milestones** (7)
- **missing features** (1)
- **mlocate** (6)
- **mobile devices** (2)
- **mount** (1)
- **mtp** (6)
- **multiboot usb** (7)
- **nccdc** (1)
- **netanalysis** (1)
- **netflow** (1)
- **ntfs** (5)
- **offensive forensics** (2)
- **office** (1)
- **outlook** (3)
- **outlook web access** (2)
- **owa** (2)
- **paladin** (1)
- **pdf** (2)
- **perl** (3)
- **persistence** (1)
- **pfic** (1)
- **posix** (4)
- **powerpoint** (1)
- **prefetch** (2)
- **re-c** (1)
- **re-creation testing** (2)
- **recon** (1)
- **redteam** (1)
- **regripper** (1)
- **remote** (1)
- **research** (1)
- **rhel** (6)
- **sample images** (1)
- **SANS** (5)
- **Saturday reading** (43)
- **scrap files** (1)
- **setmace** (1)
- **setupapi** (1)
- **shadowkit** (1)
- **shadows** (1)
- **shellbags** (2)
- **slow down** (1)
- **speed** (1)
- **sqlite** (1)
- **sunday funday** (24)
- **swgde** (2)
- **temporary files** (1)
- **timeline** (1)
- **timestamps** (4)

which the $MFT integration was already on our roadmap. Getting the full use out all of this information will require a database and were not sure if SQLLite is up to the task, hope to have something workable out there soon.

In the next blog post I'll talk about how to get access to the $USNJrnl, $MFT and $Logfile from volume shadow copies as not all access methods are equal. After that I'll likely move into updating some old 'what did they take' posts to reflect new artifact sources and post the results of our forensic tool tests.

Posted by David Cowen at **11:57 PM**
Labels: **indepth**, **ntfs**, **triforce**

5 comments

Add a comment as Lucius Riccio

Top comments

David Cowen via Google+ 1 year ago - Shared publicly

New blog post on the internals of three NTFS forensic artifacts and how they link to each other creating a new source of historical actions taken against a file or directory

+2 1 · Reply

Digital Forensic Source via Google+

1 year ago - Shared publicly

#DFIR

1 · Reply

Tom Yarrish

1 year ago - Digital Forensics and Incident Response (Discussion)

David Cowen originally shared this
New blog post on the internals of three NTFS forensic artifacts and how they link to each other creating a new source of historical actions taken against a file or directory

+3 1

Bill O'Sullivan 1 year ago

Thank you!

Anonymous 1 year ago

thanks for share.

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

G-C Partners, LLC

660 North Central Expressway
Suite 230
Plano, Texas 75074

VOICE: 214.291.7333
FAX: 214.722.0238

info@g-cpartners.com

XHTML 1.0
CSS 2.0