COMP6445

Digital Forensics

Filesystems and Timelining

We will cover:

- Disk Geometry, Volumes and Slack
- How Deleted Files in FAT32 and NTFS work
- Uncovering File Slack & Unallocated Space
- Date & Time in NTFS & FAT
- Timelining Analysis Techniques

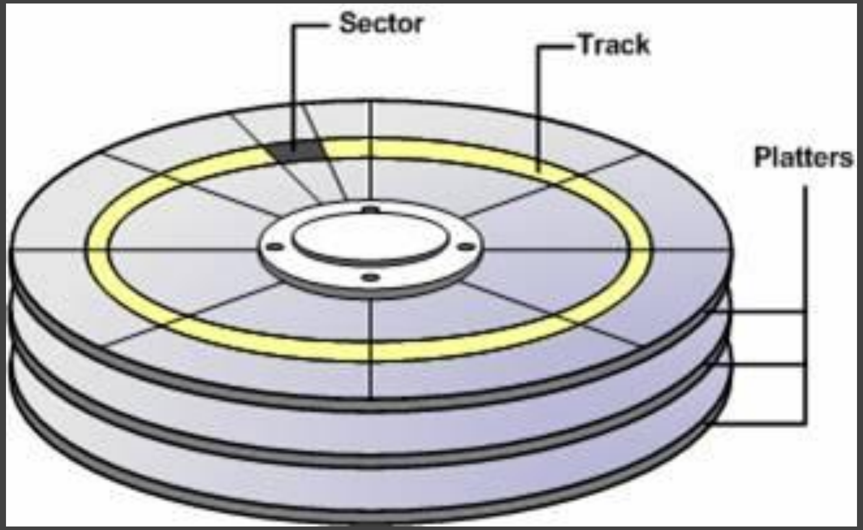
Drive Physical Structure

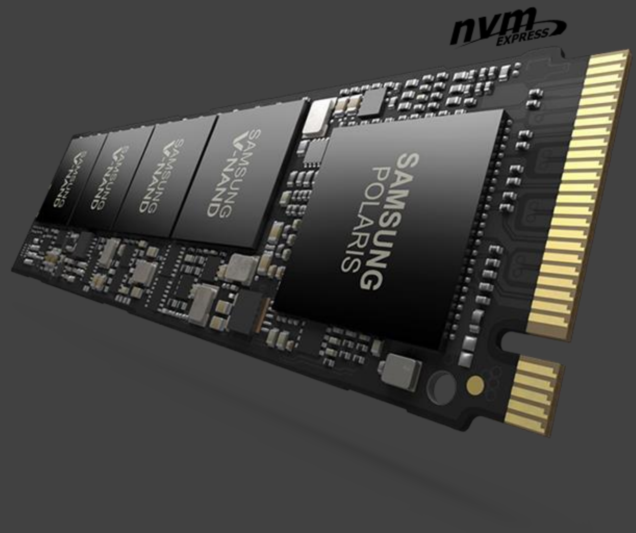
Traditional Drives are arranged into a number of circular platters, each containing a number of concentric tracks. Each track contains a number of blocks known as sectors (for older drives, 512b in size, newer drives have moved to 4096b) which are the base unit that is addressable on the disk.

We do not address these physical sectors by their Platter, Track or Sector location, but instead use a sequential Logical Block Address (LBA) scheme.

(Own research – what is CHS Addressing?)

Note it is advantageous to platter drives to store files in contiguous sectors. This has an impact on file recovery, particularly data carving.





Drive Physical Structure

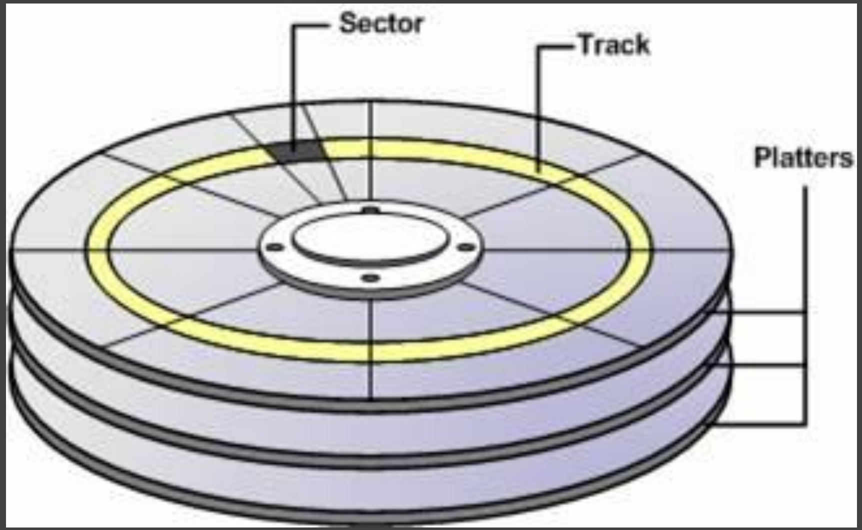
Solid State Drives mostly use NAND Flash Memory chips instead of platters, so the old concepts don't really apply.

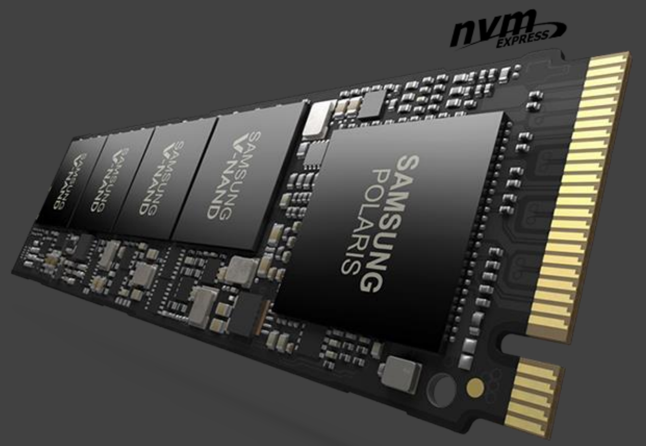
We still use an LBA addressing scheme at a logical level though. The translation to physical layer is handled by drivers/firmware.

Note there is no distinct advantage to contiguous storage on an SSD, in fact wear levelling algorithms mean that almost the exact opposite happens.

Sectors can be written in what almost looks like a random pattern.

This can make data carving for all but the smallest files very difficult.





Drive Logical Structure

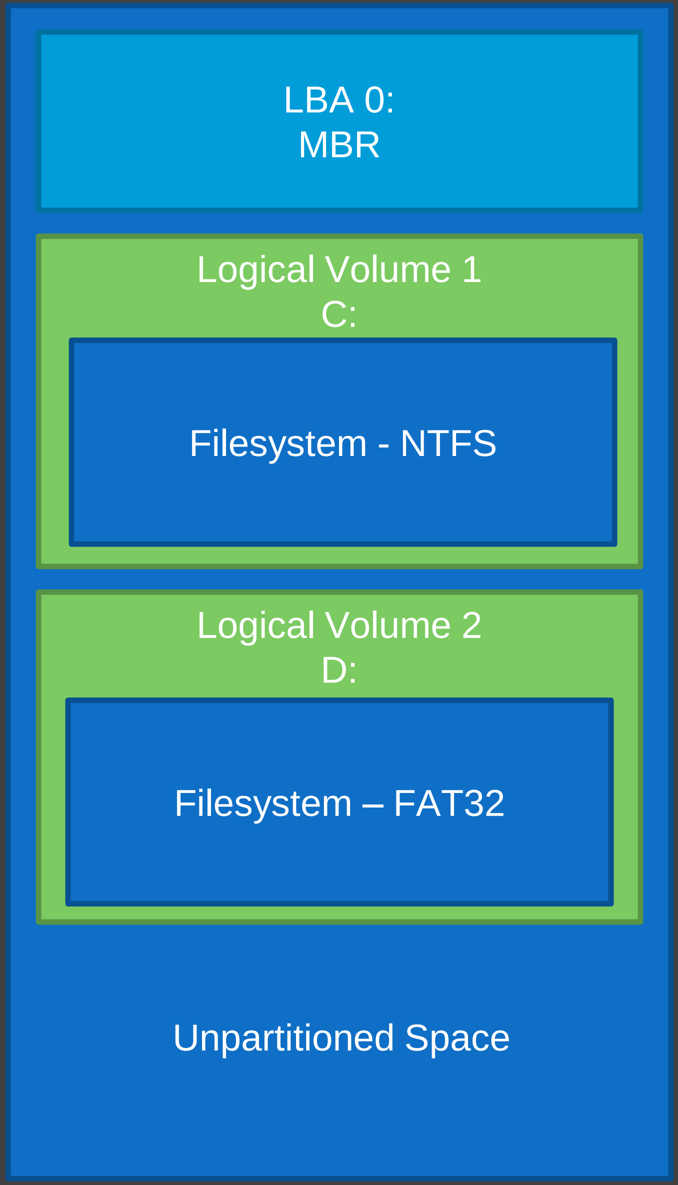
MBR – Master Boot Record, contains the volume locations and sizes.

Volumes – Logical containers for filesystems. Note that while the Filesystem and Volume are often referred to as the same thing, they are technically separate. (You can have an empty volume!)

Filesystems – Structures that sit within a volume, and allow files to be organised and saved.

Clusters – Sectors are not an efficient way for filesystems to address data blocks. Therefore, they are organised into clusters. The cluster size is how many sectors are grouped into a single cluster. Files are assigned entire clusters as they grow.

Unpartitioned Space – If you don't assign all sectors to a partition, they will sit within unpartitioned space on the drive.



Master Boot Record

Master Boot Record is the first sector on the drive, or LBA0.

First section of the MBR contains bootstrap code. If the drive is bootable, this contains the required code to flick control over into the bootable volume.

The next 64 bytes contain a Partition Table that contains a max of 4 entries.

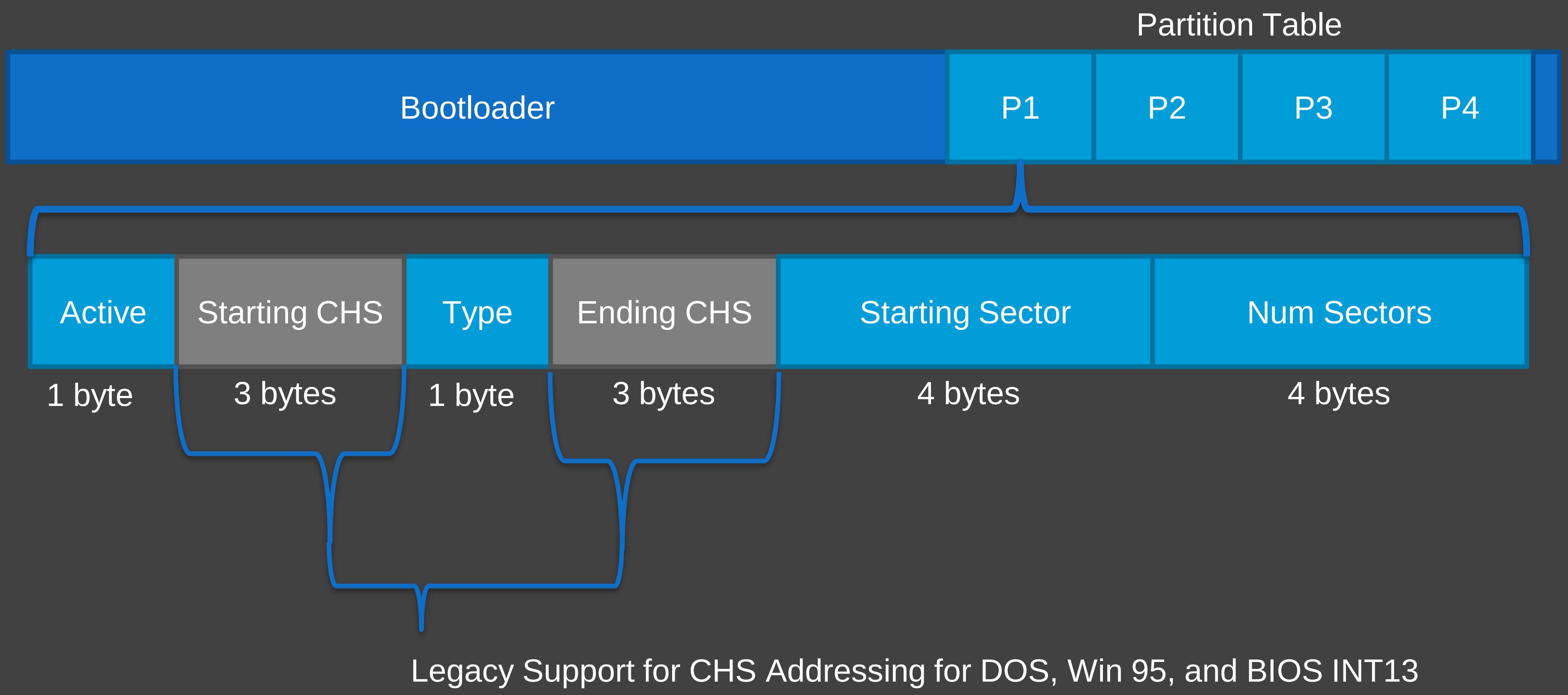
Address		Description		Size in bytes
Hex	Dec			
0000	0	**Code Area**		**≤ 446**
01B8	*440*	*Optional disk signature*		*4*
01BC	*444*	*Usually null: 0x0000*		*2*
01BE	446	**Table of primary partitions** (four 16-byte partition structures)		**64**
01FE	510	55h	**MBR signature**: 0xAA55	**2**
01FF	511	AAh		
MBR total size: 446 + 64 + 2 =				**512**

Drawback of MBR:

- Only natively supported 4 volumes, though extended volumes were introduced to support more.
- Original CHS addressing method used in initial partition table implementation.
- Size limitation due to use of 32bit sector addresses in partition table.

MBR Partition Table

Each Partition Entry Describes the start sector, size, and partition type.



GUID Partition Table

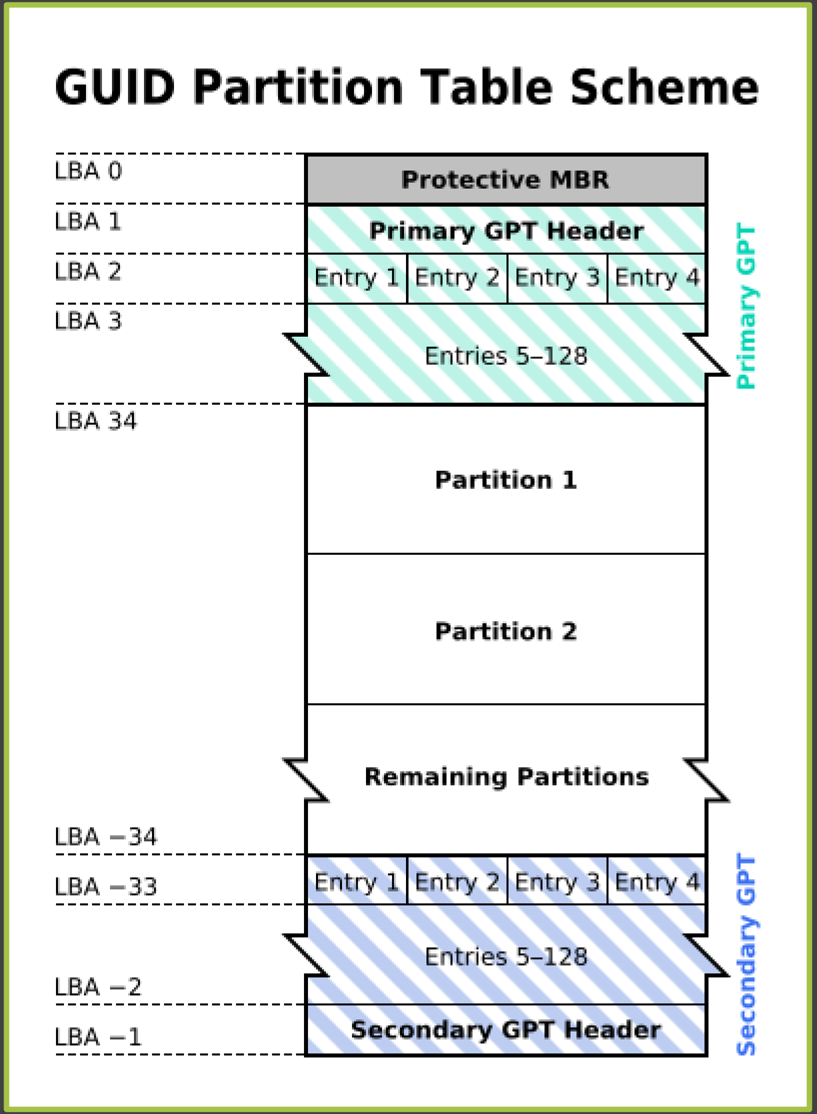
Contains a “protective MBR” at LBA0 to ensure older devices don’t wipe the drive on connect. It will contain a single partition entry with type code 0xEEh

Much more space is assigned to the partition table to allow for up to 128 partitions, with a max capacity of 18 exabytes.

At a min LBA2-34 are assigned to the partition table.

GUID partition entry format

Offset	Length	Contents
0 (0x00)	16 bytes	Partition type GUID
16 (0x10)	16 bytes	Unique partition GUID
32 (0x20)	8 bytes	First LBA (little endian)
40 (0x28)	8 bytes	Last LBA (inclusive, usually odd)
48 (0x30)	8 bytes	Attribute flags (e.g. bit 60 denotes read-only)
56 (0x38)	72 bytes	Partition name (36 UTF-16LE code units)

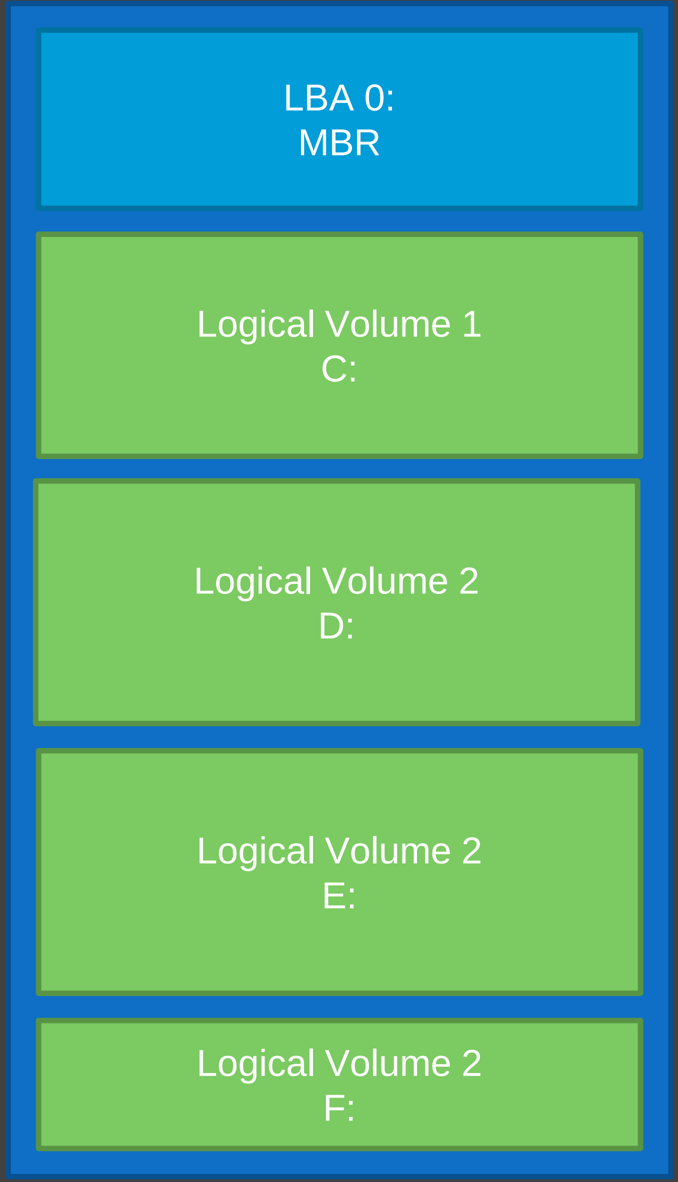


Modifying the Partition Table

What happens if we play with a partition table?

Active	Starting CHS	Type	Ending CHS	Starting Sector	Num Sectors
80h	…	07h	…	LBA 1	10
00h	…	00h	…	00h	00h
80h	…	07h	…	LBA 41	10
80h	…	07h	…	LBA 51	5

All of the partition data still exists, the OS just does not recognise that a partition exists in that location. We can read data by restoring the partition table, or directly access the clusters we know to contain the data.



Unpartitioned Space

Can we get data there?

- Raw Disk Editor to make manual changes – WinHex
- OS APIs generally don't like writing to "invalid" spaces on disk.
- Driver level APIS.

What could we store?

- Potentially, whatever fits!

Small amounts of unpartitioned can be normal, but remember, large amounts of unparitioned space should stick out like a sore thumb to investigators.

Partition	Start LBA	Sector Count
Part-1	1	10,000
Part-2	20,001	1,000,000
Part-3	1,020,001	2,000,000
		3,010,000

- Empty gap in the partitions between LBA 10,001 and LBA 20,000
- Sum of all partitions sectors is less than the total drive sectors. (Remember to include MBA/GPT in this calculation)

Hidden Partitions

Using a tool to modify / restores the partition table, one could hide/unhide partitions.

On its own this would be easy to detect…

- The partition table may retain traces of tampering if not done correctly.
- Viewing the unpartitioned space directly would show the data in plain text.
- Most filesystems have a tell-tale signature at the start of the volume known as the Volume Boot Record. One might search for common signatures across unpartitioned space in search of unlisted partitions.

What if, you encrypt the partition?

- Data looks random to the naked eye.
- Filesystem signatures would not be visible to scanning tools.
- There is still a big gap in the partition table that would seem suspicious.

If your interested in more advanced partition hiding techniques, Google “VeraCrypt Hidden Volume”

Volume Slack

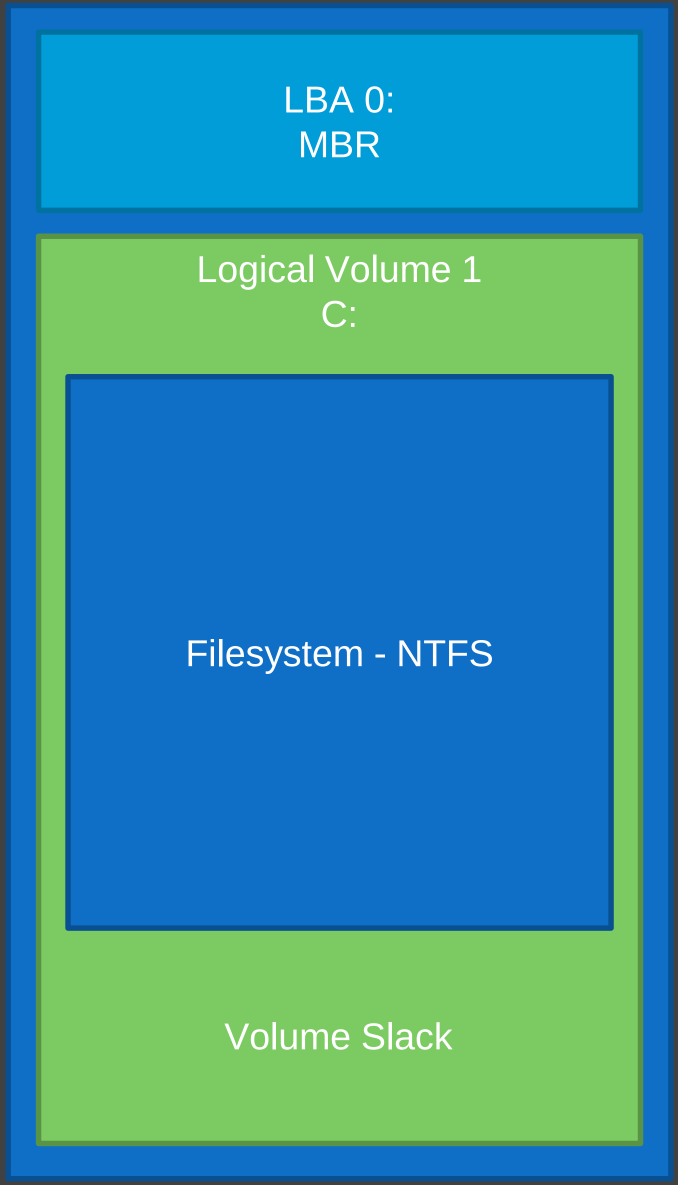
Volume slack is the unused space between the end of file system and end of the partition where the file system resides - also defined as sectors at the end of the volume or partition that cannot be allocated to a cluster.

This happens when the partition size is not multiple of the cluster size.

For example, lets assume create an partition that is 100 sectors. You create an NTFS filesystem in that partition that has cluster size of 15 sectors per cluster.

100 / 15 = 6 and remainder 10. – These 10 sectors are not used assigned to a cluster in the filesystem, and are unused.

10 * 512b (sector size) = 5Kb slack space.



Summary – Hiding Techniques So Far….

- Writing Directly to Unpartioned Space
- Deleted Volumes and Hidden Partitions
- Hiding in Volume Slack

Filesystem Basics

- Filesystems are the way in which operating systems manage files and folder structures on the disk.
- They are another logical layer of abstraction on top of volumes.
- The also manage the allocation of clusters to files, and track the usage of clusters (ie which are used, and which are free for use)
- There are a large number of filesystems, we will look in depth at FAT32 and NTFS as the most prevalent.
- Its important to have abase level of knowledge in other filesystems, as you never know what you might come across.

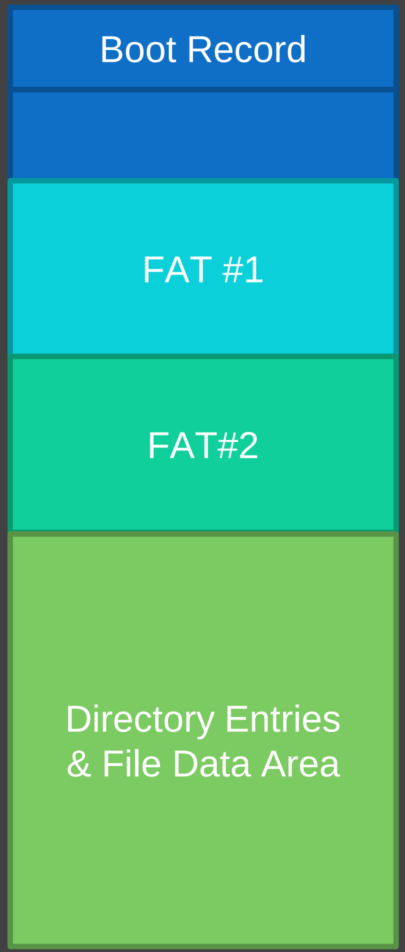
FAT32 Overview

- FAT32 was originally developed for MS DOS 7.0 / Win 95.

- Originally developed out of FAT8, FAT12, and FAT16, with the concept being essentially the same across all, just the number of bytes used to address clusters increasing with each revision to support larger sizes.

- FAT32 is still commonly used today, primarily on USB sticks and SD Cards for compatibility reasons.

- 3 Main components of the FAT
- Boot Record
- File Allocation Table (FAT)
- Directory Entries



Volume Boot Record

Overview of data stored in VBR:

- Jump to boot code
- OEM Name – Often MSDOS
- BPB - BIOS Parameter Block - Database of values that setup the FAT.
- Boot Code and Error Messages

BPB Contains:

- File system geometry (bytes per sector, sectors per cluster)
- Number of FATs stored.
- Location of the root directory (typically Cluster 2)
- Volume Label in ASCII
- Type Label - ‘FAT32’ in ASCII

```
00000000 EB 58 90 4D 53 44 4F 53-35 2E 30 00 02 08 CE 10 ëX·MSDOS5.0···Î·
00000010 02 00 00 00 00 F8 00 00-3F 00 FF 00 F5 00 00 00 ·····ø··?·ÿ·õ···
00000020 0B 83 1E 00 99 07 00 00-00 00 00 00 02 00 00 00 ················
00000030 01 00 06 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
00000040 80 00 29 B7 B7 7D 66 4E-4F 20 4E 41 4D 45 20 20 ··)··}fNO NAME
00000050 20 20 46 41 54 33 32 20-20 20 33 C9 8E D1 BC F4   FAT32   3É·Ñ¼ô
00000060 7B 8E C1 8E D9 BD 00 7C-88 56 40 88 4E 02 8A 56 {·Á·Ù½·|·V@·N··V
00000070 40 B4 41 BB AA 55 CD 13-72 10 81 FB 55 AA 75 0A @´A»ªUÍ·r··ûUªu·
00000080 F6 C1 01 74 05 FE 46 02-EB 2D 8A 56 40 B4 08 CD öÁ·t·þF·ë-·V@´·Í
00000090 13 73 05 B9 FF FF 8A F1-66 0F B6 C6 40 66 0F B6 ·s·¹ÿÿ·ñf·¶Æ@f·¶
000000a0 D1 80 E2 3F F7 E2 86 CD-C0 ED 06 41 66 0F B7 C9 Ñ·â?÷â·ÍÀí·Af··É
000000b0 66 F7 E1 66 89 46 F8 83-7E 16 00 75 39 83 7E 2A f÷áf·Fø·~··u9·~*
000000c0 00 77 33 66 8B 46 1C 66-83 C0 0C BB 00 80 B9 01 ·w3f·F·f·À·»··¹·
000000d0 00 E8 2C 00 E9 A8 03 A1-F8 7D 80 C4 7C 8B F0 AC ·è,·é¨·¡ø}·Ä|·ð¬
000000e0 84 C0 74 17 3C FF 74 09-B4 0E BB 07 00 CD 10 EB ·Àt·<ÿt·´·»··Í·ë
000000f0 EE A1 FA 7D EB E4 A1 7D-80 EB DF 98 CD 16 CD 19 î¡ú}ëä¡}·ëß·Í·Í·
00000100 66 60 80 7E 02 00 0F 84-20 00 66 6A 00 66 50 06 f`·~····  ·fj·fP·
00000110 53 66 68 10 00 01 00 B4-42 8A 56 40 8B F4 CD 13 Sfh····´B·V@·ôÍ·
00000120 66 58 66 58 66 58 66 58-EB 33 66 3B 46 F8 72 03 fXfXfXfXë3f;Før·
00000130 F9 EB 2A 66 33 D2 66 0F-B7 4E 18 66 F7 F1 FE C2 ùë*f3Òf··N·f÷ñþÂ
00000140 8A CA 66 8B D0 66 C1 EA-10 F7 76 1A 86 D6 8A 56 ·Êf·ÐfÁê·÷v··Ö·V
00000150 40 8A E8 C0 E4 06 0A CC-B8 01 02 CD 13 66 61 0F @·èÀä··Ì¸··Í·fa·
00000160 82 74 FF 81 C3 00 02 66-40 49 75 94 C3 42 4F 4F ·tÿ·Ã··f@Iu·ÃBOO
00000170 54 4D 47 52 20 20 20 20-00 00 00 00 00 00 00 00 TMGR    ········
00000180 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
00000190 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
000001a0 00 00 00 00 00 00 00 00-00 00 00 00 0D 0A 44 69 ··············Di
000001b0 73 6B 20 65 72 72 6F 72-FF 0D 0A 50 72 65 73 73 sk errorÿ··Press
000001c0 20 61 6E 79 20 6B 65 79-20 74 6F 20 72 65 73 74  any key to rest
000001d0 61 72 74 0D 0A 00 00 00-00 00 00 00 00 00 00 00 art·············
000001e0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
000001f0 00 00 00 00 00 00 00 00-AC 01 B9 01 00 00 55 AA ········¬·¹···Uª
00000200 52 52 61 41 00 00 00 00-00 00 00 00 00 00 00 00 RRaA············
```

This has a very distinctive look in Hex/Text, and is easy to see how we could scan for deleted or orphaned filesystems/volumes.

Directory Entries

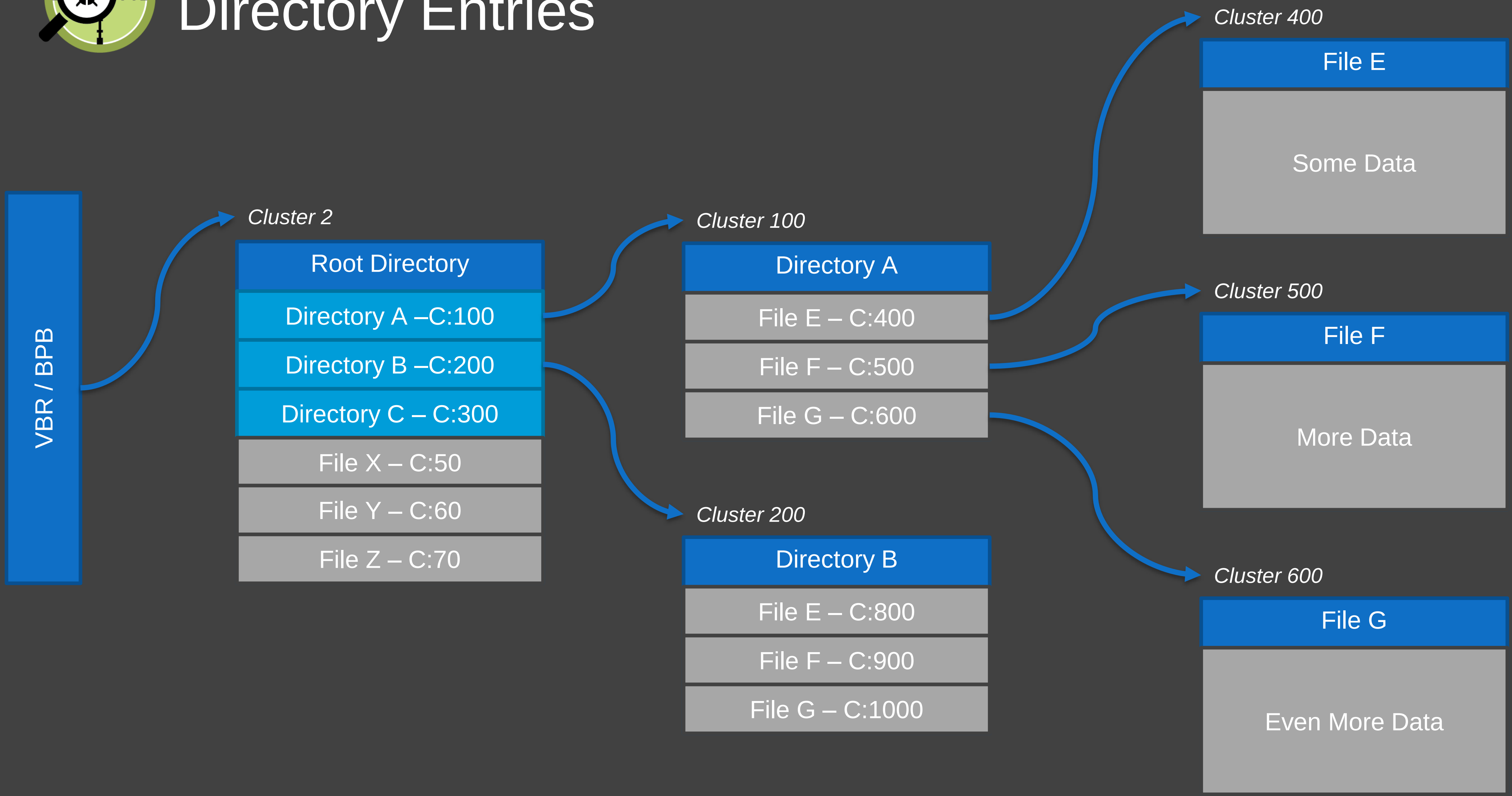
- File information in FAT systems are organized into Directory Entry structures. Starting with the Root Directory. A pointer to the Root Directory is found in the Boot Sector.

- Each Directory entry consists of file metadata, including pointers to the clusters that contain data for the file/folder. For files, the clusters contain the content of the file. For folders, the clusters contain another directory entry.

- If attempting to parse entries manually, remember this is all stored in little endian.

FAT32 - Directory Entry
Filename
Extension
Attributes
Reserved
Create Time
Create Date
Last Access Date
First Cluster (Most Sig. Bits)
Last Mod Time
Last Mod Date
First Cluster (Least Sig. Bits)
File Size

Directory Entries



File Allocation Table

Directory entries only store the beginning cluster a file. They don't track the additional clusters that get used beyond the first.

The File Allocation Table tracks the status of all clusters in the file system. It is used as both a way of determining all clusters in use by a particular file, and also as a global tracker of what clusters are in use, and what are free for allocation to a file.

It is basically great big linked list, with each value in the FAT pointing to the next cluster used by the file.

Each entry in the FAT can be one of the following:

- 0x?0000000 – Indicates a free cluster, that is ready to be allocated to a file.
- 0x?0000002 to 0x?FFFFFEF – Indicates the cluster is in use. The value stored points to the next cluster of the file.
- 0x?FFFFFFF – Indicates the cluster is in use, however there are no more clusters used by the file (EOF Marker)

File Allocation Table Example

Directory A
File A – C:007
File B – C:031
File C – C:034
File D – C:03A

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	FF0	FFF	000	000	000	000	000	008	009	00A	00B	00C	020	00E	FFF	000
10	000	000	000	000	045	046	047	048	049	04A	04B	04C	04D	04E	04F	FFF
20	021	022	023	024	025	026	027	054	000	000	000	000	000	000	000	000
30	031	032	037	034	035	036	050	038	039	03E	03B	03C	03D	FFF	03F	040
40	041	FFF	000	000	000	000	000	000	000	000	000	000	000	000	000	000
50	051	052	053	00D	055	056	057	058	059	05A	05B	05C	FFF	000	000	000

File Allocation Table Example

Directory A
File A – C:007
File B – C:031
File C – C:034
File D – C:03A

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	FF0	FFF	000	000	000	000	000	008	009	00A	00B	00C	020	00E	FFF	000
10	000	000	000	000	015	016	017	018	019	01A	01B	01C	01D	01E	01F	FFF
20	021	022	023	024	025	026	027	054	000	000	000	000	000	000	000	000
30	031	032	037	034	035	036	050	038	039	03E	03B	03C	03D	FFF	03F	040
40	041	FFF	000	000	000	000	000	000	000	000	000	000	000	000	000	000
50	051	052	053	00D	055	056	057	058	059	05A	05B	05C	FFF	000	000	000

Deleted Files – FAT32

When a file is deleted, a few things happen:

1. The first character of the filename attribute is modified to 0xe5 (often graphically represented with '_' or sometimes '!" in forensics tools).
2. All cluster runs in the FAT are replaced with 0x00 to return them to available state.

Implications for recovery:

1. Deleted files directory entries can easily identified by scanning for entries starting with 0xe5
2. The location of the starting cluster is retained in the deleted entry.
3. The file size is retained in the deleted entry.
4. Cluster runs are zeroed out in the FAT, so we wont be able to follow them to restore data. We can make educated guessed as to their location, based on our knowledge that allocation algorithms want to keep files as contiguous an unfragmented as possible.

Deleted File Example - Simple

We can use the remaining information in the Directory Entry, along with knowledge of how cluster runs work, to attempt to recover the file.

We know where is started, and we know the how many clusters were in use from the file size attribute.

Directory A
File A – C:007
File B – C:031
File C – C:034
_ile D – C:03A

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	FF0	FFF	000	000	000	000	000	008	009	00A	00B	00C	020	00E	FFF	000
10	000	000	000	000	015	016	017	018	019	01A	01B	01C	01D	01E	01F	FFF
20	021	022	023	024	025	026	027	054	000	000	000	000	000	000	000	000
30	031	032	037	034	035	036	050	038	039	03E	000	000	000	000	03F	040
40	041	FFF	000	000	000	000	000	000	000	000	000	000	000	000	000	000
50	051	052	053	00D	055	056	057	058	059	05A	05B	05C	FFF	000	000	000

Deleted File Example – Not as Simple

In this case, if we just use the starting cluster and file size, we run into another allocated file.

We can make an informed guess, and move to the next set of unallocated clusters.

Directory A
File A – C:007
_ile B – C:030
File C – C:034
File D – C:03A

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	FF0	FFF	000	000	000	000	000	008	009	00A	00B	00C	020	00E	FFF	000
10	000	000	000	000	015	016	017	018	019	01A	01B	01C	01D	01E	01F	FFF
20	021	022	023	024	025	026	027	054	000	000	000	000	000	000	000	000
30	000	000	000	034	035	036	050	000	000	000	03B	03C	03D	FFF	000	000
40	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
50	051	052	053	00D	055	056	057	058	059	05A	05B	05C	FFF	000	000	000

Deleted File Example – Problem

File C jumps forward into the middle of an unallocated cluster run and then back across multiple cluster runs.

This makes it almost impossible to even make an informed guess as to how the clusters were allocated.

File Fragmentation has an impact on FAT32 file recovery.

Directory A
File A – C:007
File B – C:031
_ile C – C:034
File D – C:03A

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	FF0	FFF	000	000	000	000	000	000	009	00A	00B	00C	020	000	000	000
10	000	000	000	000	015	016	017	018	019	01A	01B	01C	01D	01E	01F	FFF
20	021	022	023	024	025	026	027	054	000	000	000	000	000	000	000	000
30	031	032	037	000	000	000	000	038	039	03E	03B	03C	03D	FFF	03F	040
40	041	FFF	000	000	000	000	000	000	000	000	000	000	000	000	000	000
50	000	000	000	000	055	056	057	058	059	05A	05B	05C	FFF	000	000	000

NTFS Overview

Introduced with Windows NT 3.1, it has been the default for Windows Systems since that time.

NFTS introduces a number of new features

- sparse file support
- disk use quotas
- reparse points
- distributed link tracking
- file-level encryption (EFS)

In NTFS, everything is a file, even the structures that make it up can be referenced by a file.

There is a lot more to NTFS from a forensic perspective than just the $MFT and deleted files, particularly the journaling aspect of the log file that we wont cover today.

NTFS VBR

Contains similar types of information to the FAT VBR

- Jump instruction
- OEM ID “NTFS”
- Filesystem Geometry
- Cluster that contains the $MFT
- Cluster that contains the backup $MFT ($MFTMirr)

Once again, visual inspection shows this to standout immediately to an observer.

```
000000000 EB 52 90 4E 54 46 53 20-20 20 20 00 02 08 00 00 ëR·NTFS     ·····
000000010 00 00 00 00 00 F8 00 00-3F 00 FF 00 00 28 03 00 ·····ø··?·ÿ··(··
000000020 00 00 00 00 80 00 80 00-FF CF 7C 02 00 00 00 00 ········ÿÏ|·····
000000030 00 00 0C 00 00 00 00 00-02 00 00 00 00 00 00 00 ················
000000040 F6 00 00 00 01 00 00 00-48 7A 0C CA 8D 0C CA C8 ö·······Hz·Ê··ÊÈ
000000050 00 00 00 00 FA 33 C0 8E-D0 BC 00 7C FB 68 C0 07 ····ú3À·Ð¼·|ûhÀ·
000000060 1F 1E 68 66 00 CB 88 16-0E 00 66 81 3E 03 00 4E ··hf·Ë····f·>··N
000000070 54 46 53 75 15 B4 41 BB-AA 55 CD 13 72 0C 81 FB TFSu·´A»ªUÍ·r··û
000000080 55 AA 75 06 F7 C1 01 00-75 03 E9 DD 00 1E 83 EC Uªu·÷Á··u·éÝ···ì
000000090 18 68 1A 00 B4 48 8A 16-0E 00 8B F4 16 1F CD 13 ·h··´H·····ô··Í·
0000000a0 9F 83 C4 18 9E 58 1F 72-E1 3B 06 0B 00 75 DB A3 ··Ä··X·rá;···uÛ£
0000000b0 0F 00 C1 2E 0F 00 04 1E-5A 33 DB B9 00 20 2B C8 ··Á.····Z3Û¹· +È
0000000c0 66 FF 06 11 00 03 16 0F-00 8E C2 FF 06 16 00 E8 fÿ········Âÿ···è
0000000d0 4B 00 2B C8 77 EF B8 00-BB CD 1A 66 23 C0 75 2D K·+Èwï¸·»Í·f#Àu-
0000000e0 66 81 FB 54 43 50 41 75-24 81 F9 02 01 72 1E 16 f·ûTCPAu$·ù··r··
0000000f0 68 07 BB 16 68 70 0E 16-68 09 00 66 53 66 53 66 h·»·hp··h··fSfSf
000000100 55 16 16 16 68 B8 01 66-61 0E 07 CD 1A 33 C0 BF U···h¸·fa··Í·3À¿
000000110 28 10 B9 D8 0F FC F3 AA-E9 5F 01 90 90 66 60 1E (·¹Ø·üóªé_···f`·
000000120 06 66 A1 11 00 66 03 06-1C 00 1E 66 68 00 00 00 ·f¡··f······fh···
000000130 00 66 50 06 53 68 01 00-68 10 00 B4 42 8A 16 0E ·fP·Sh··h··´B···
000000140 00 16 1F 8B F4 CD 13 66-59 5B 5A 66 59 66 59 1F ····ôÍ·fY[ZfYfY·
000000150 0F 82 16 00 66 FF 06 11-00 03 16 0F 00 8E C2 FF ····fÿ········Âÿ
000000160 0E 16 00 75 BC 07 1F 66-61 C3 A0 F8 01 E8 09 00 ···u¼··faÃ ø·è··
000000170 A0 FB 01 E8 03 00 F4 EB-FD B4 01 8B F0 AC 3C 00  û·è··ôëý´··ð¬<·
000000180 74 09 B4 0E BB 07 00 CD-10 EB F2 C3 0D 0A 41 20 t·´·»··Í·ëòÃ··A 
000000190 64 69 73 6B 20 72 65 61-64 20 65 72 72 6F 72 20 disk read error 
0000001a0 6F 63 63 75 72 72 65 64-00 0D 0A 42 4F 4F 54 4D occurred···BOOTM
0000001b0 47 52 20 69 73 20 6D 69-73 73 69 6E 67 00 0D 0A GR is missing···
0000001c0 42 4F 4F 54 4D 47 52 20-69 73 20 63 6F 6D 70 72 BOOTMGR is compr
0000001d0 65 73 73 65 64 00 0D 0A-50 72 65 73 73 20 43 74 essed···Press Ct
0000001e0 72 6C 2B 41 6C 74 2B 44-65 6C 20 74 6F 20 72 65 rl+Alt+Del to re
0000001f0 73 74 61 72 74 0D 0A 00-8C A9 BE D6 00 00 55 AA start····©¾Ö··Uª
```

$MFT & $Bitmap

- The main component of NTFS as it relates to Files is the Master File Table ($MFT). This is the equivalent of "Directory Entries" in FAT. Every file or folder must have its own record in the $MFT, which contains metadata about the file, and the location of the data itself.

- The $MFT also has an attribute that keeps track of which MFT FILE entries are in-use, and which are free for reuse. This is known as the MFT Entry Bitmap, but is not to be confused with the $BITMAP file.

- The $BITMAP is the equivalent of the FAT, and tracks the usage of clusters in the filesystem. It is much more efficient than a FAT, as it tracks each cluster with a single bit (in-use or free). Unlike the FAT it does not track the next cluster in the file.

$MFT		
File 1	Attributes & Data	…
File 2	Attributes & Data	…
File 3	Attributes & Data	…
File 4	Attributes & Data	…
…		

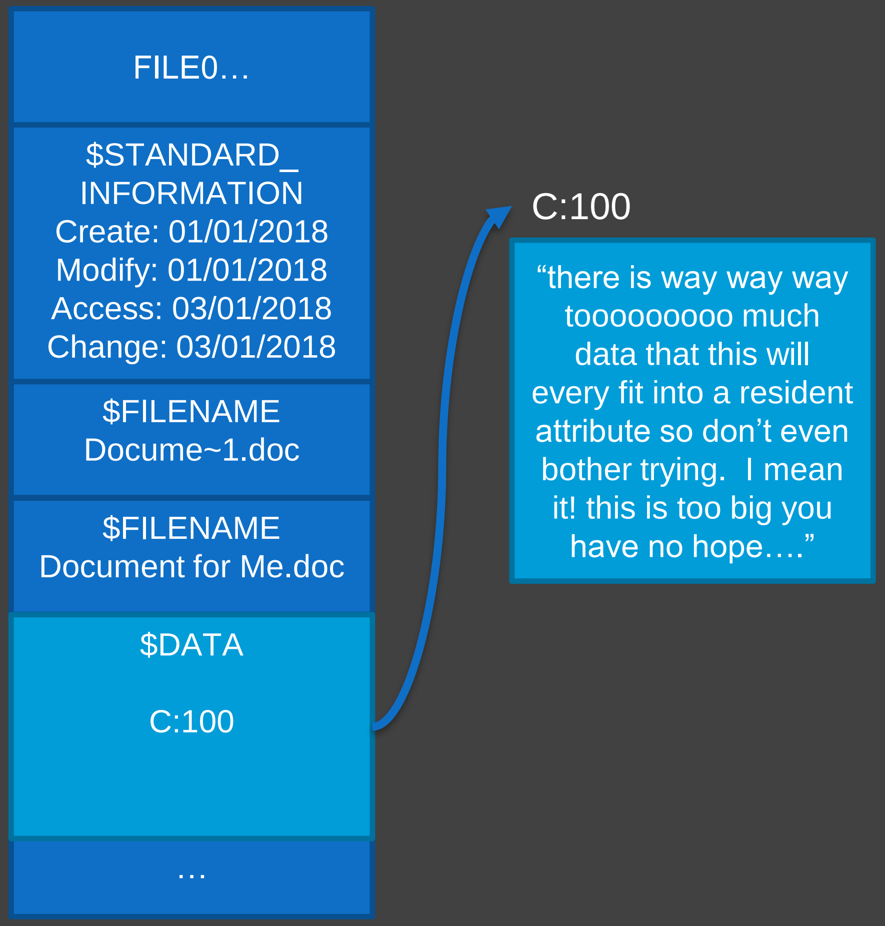
$BITMAP
01010100010100010010111110100100100001000100100 11000001011101101010001010010001000110111111110 01010100010100010010111110100100100001000100100 11000001011101101010001010010001000110111111110 01010100010100010010111110100100100001000100100 11000001011101101010001010010001000110111111110

FILE Records

Each file has its own FILE Record within the $MFT (even the $MFT itself). These records contain a small MFT header, and then consist of attributes which contain metadata about the file.

Key Attributes to be aware of initially:
$STANDARD_INFORMATION – Creation, Modification, Changed and Access Times
$FILENAME – The Filename (may be more than one to support 8.3 and long filenames)
$DATA – The actual content of the file. (Remember there can be more than 1 as well for ADS!)

Attribute data initially starts ‘resident’ inside the entry itself. If it grows to large to remain, it is instead moved out into the data area of the disk, and replaced in the the by clusters.



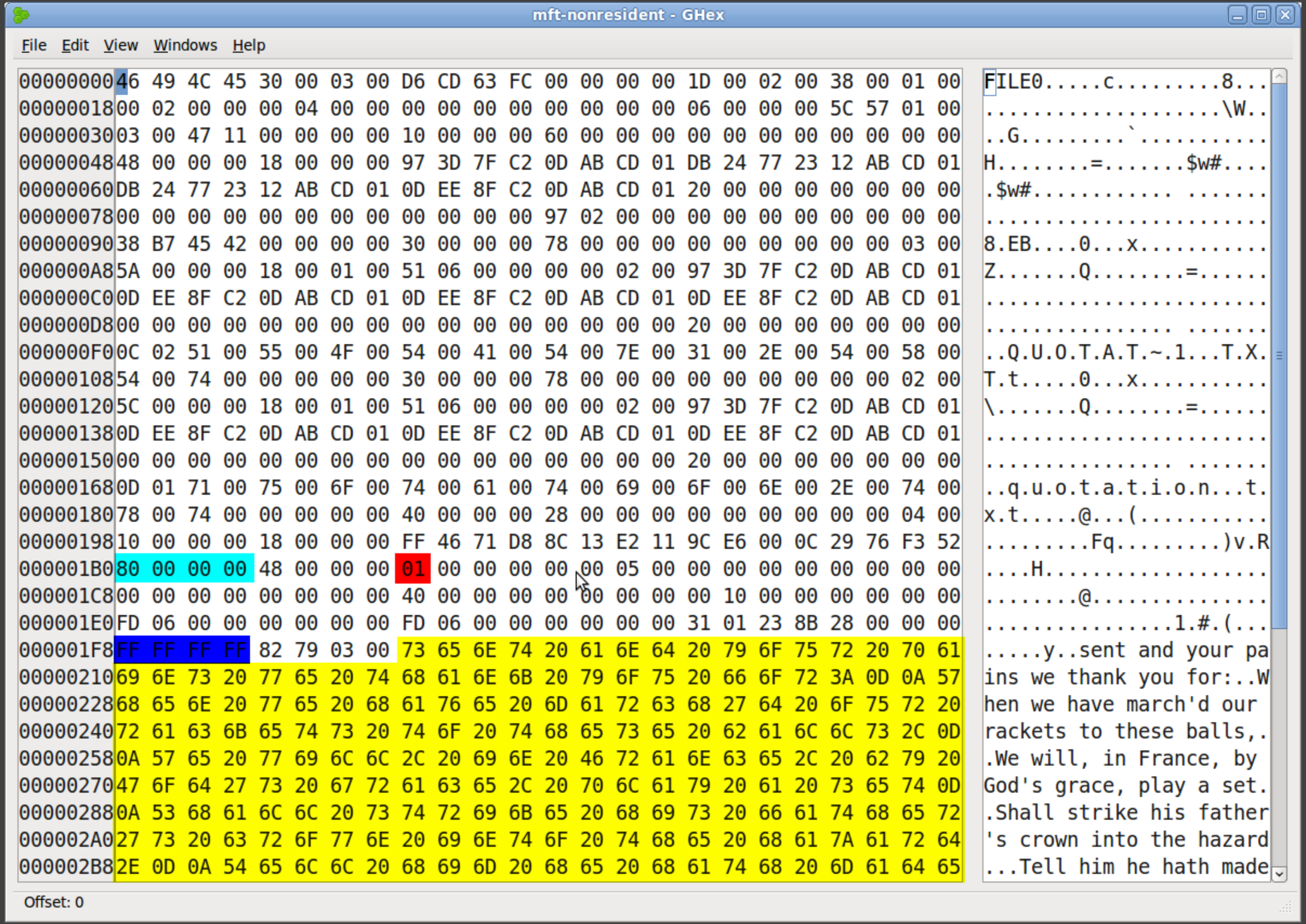
A Quick Word About Resident Data…

- If a file is initially resident, but then grows sufficiently that it needs to become non-resident, what happens to the original resident data?

- Answer, it may be partially overwritten, but it is not zeroed out.

- Attributes also never become resident again, once they become non-resident. So even overwriting the data will not touch this.

- This can be another location to recover deleted data.



$Data Attribute & Cluster Runs

- $Data attributes store the location of files using cluster runs.
- Each run consists of a header, the offset of starting cluster location and number of contiguous clusters that follows it. If the file is fragmented, additional runs are stored.
- Header - single byte which describes the number of bytes used by the following length and offset fields. Cluster runs can be variable size if additional bytes are required to describe large offsets or lengths. A header of 0x00 means no further runs.
- Length – The number of contiguous clusters in this run.
- Offsett – The offset (in clusters) from the starting point of the previous cluster run (or from 0 if first data run)

$DATA

H: 0x11 L: 0x30 O: 0x60
H: 0x21 L: 0x10 O: 0x0100
H: 0x00

Run 1:
Header = 0x11 - 1 byte length, 1 byte offset
Length = 0x30
Offset = 0x60
Run 2:
Header = 0x21 - 1 byte length, 2 byte offset
Length = 0x10
Offset = 0x0160 (0x0100 relative to 0x60)
Run 3:
Header = 0x00 - the end

File Data is Stored at:
0x30 Clusters Starting at Cluster 0x60
0x10 Clusters Starting at Cluster 0x160

Deleted Files - NTFS

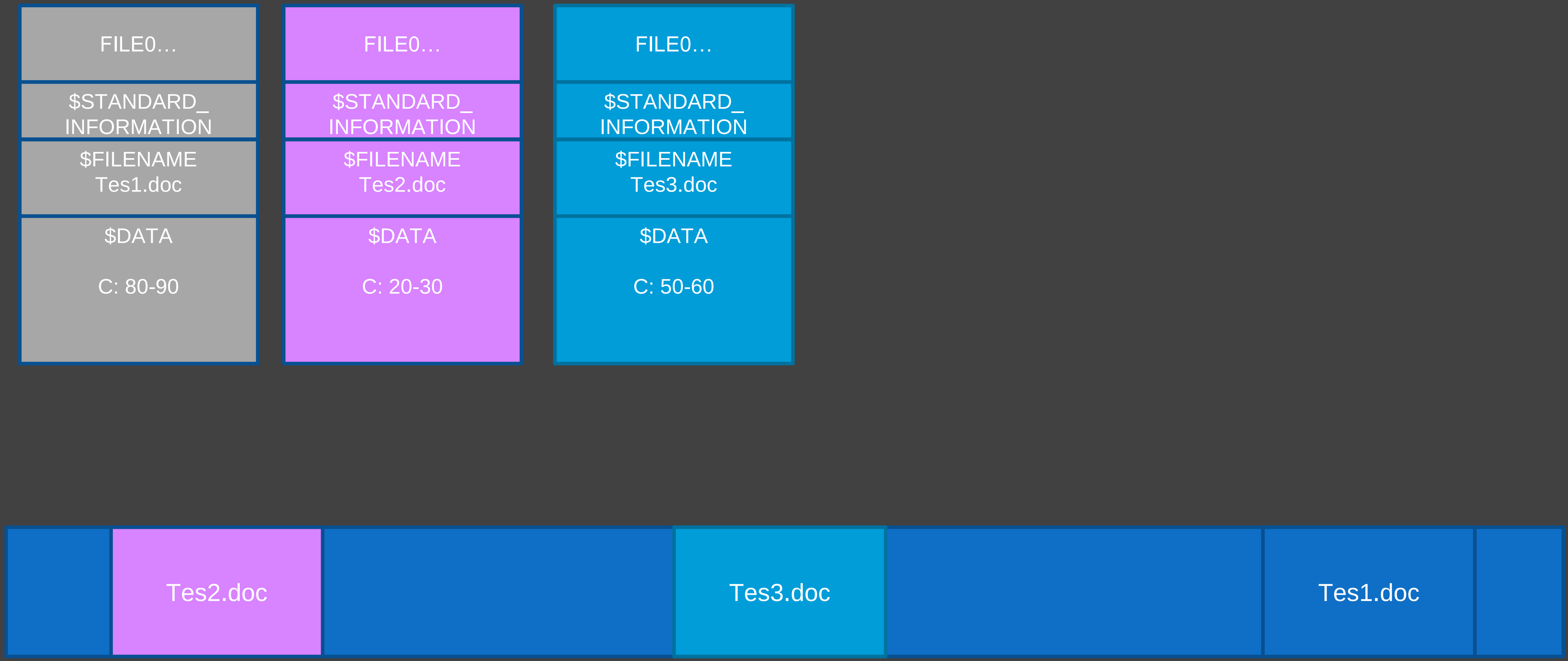
When a file is deleted in NTFS, there is a lot more tasks that occur due to additional journaling and indexing features. At a high level though, the following occurs:-

1. The MFT marks the FILE entry as available for re-use in its tracking attribute.
2. The $DATA attribute of the FILE entry is read, and the $BITMAP is updated to show that the cluster runs are no longer in use.
3. Nothing is actually wiped or deleted from either the MFT or the clusters holding data.

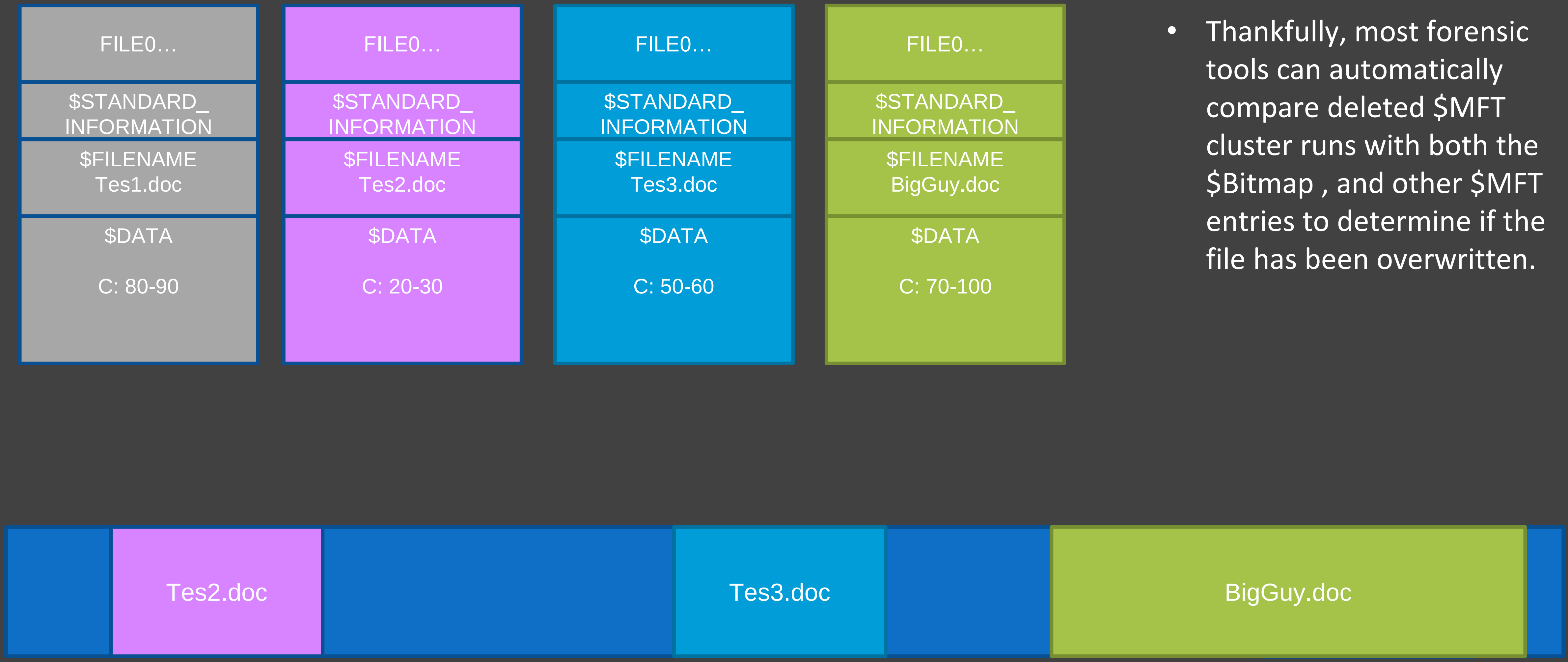
Critically, this means that:

- Until the FILE entry is overwritten, the full location of the data is stored in the $DATA attributes cluster runs. Unlike FAT, we have the starting position and length of each run.
- The data itself is still sitting in the data clusters until they are re-allocated by the system to a new file.

Deleted Files Example – Deleted Recoverable



Deleted Files Example – Deleted Overwritten



- Thankfully, most forensic tools can automatically compare deleted $MFT cluster runs with both the $Bitmap , and other $MFT entries to determine if the file has been overwritten.

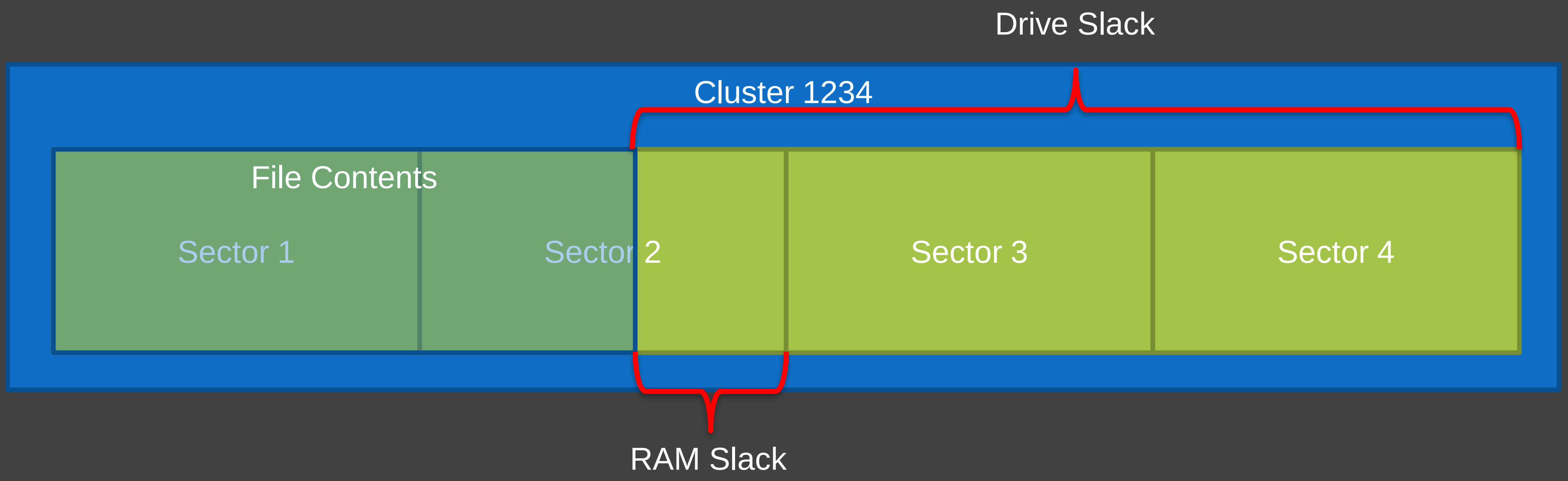
File Slack – Drive Slack and RAM Slack

- File Slack Occurs because data can only be allocated to files at the Cluster level. If a file does not fill the entire cluster, the remaining "slack" may contain residual or hidden data. This is also known as Drive Slack.

- There is a second kind of File Slack called "RAM" slack. This occurs because within each cluster, data can only be allocated in sectors.

- The operating system controls how to handle these areas. In windows, RAM slack is padded with zeros, while Drive Slack is not touched.

- Potential places to hide data, or find residual data.



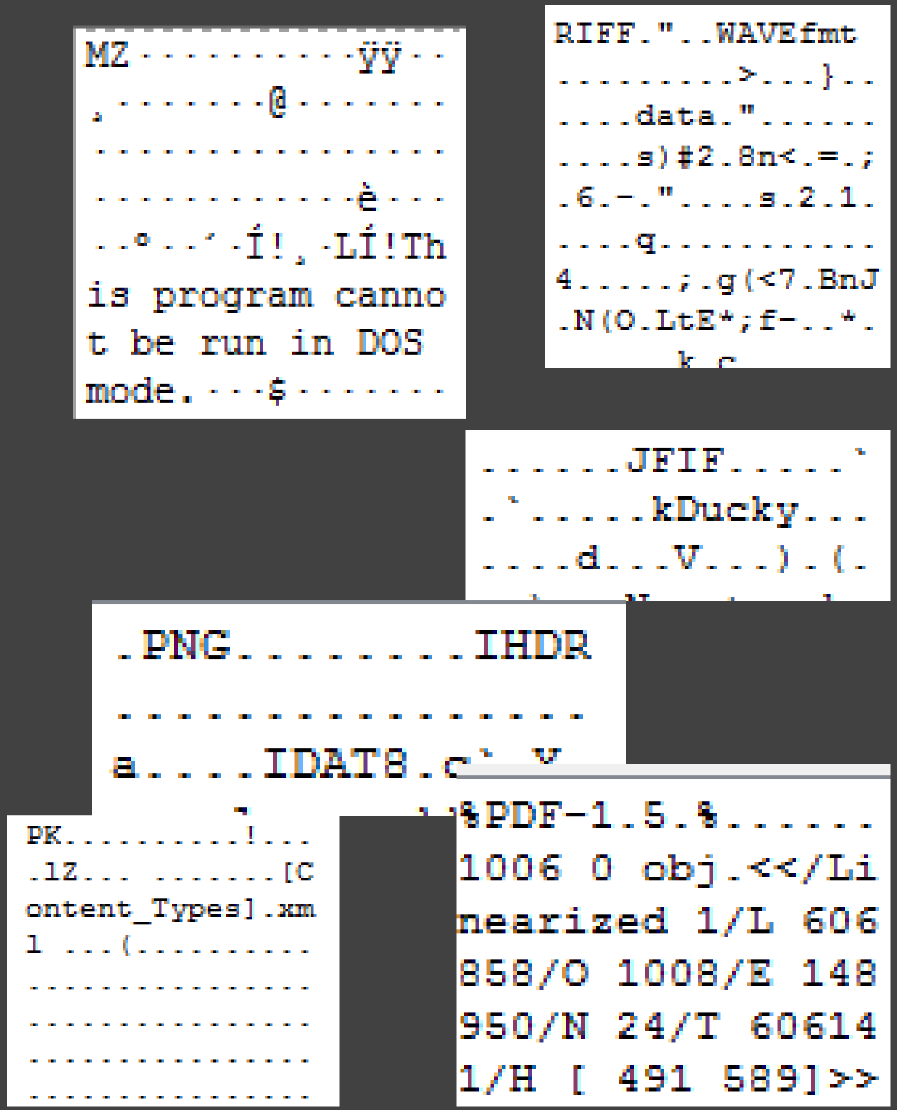
Unallocated Clusters

Unallocated space refers to the collection of clusters that are not currently assigned to any file.

So far we have talked about situations where we have a FAT entry, or a MFT entry to give us some indication of where the file was. However these is potentially many files, or file fragments that still exist in unallocated space even after there MFT or FAT entry has been overwritten and reused.

The technique is known as “File Carving”. File carving scans all clusters that are unallocated for traces of files that once existed in that space.

It does this by looking for ”File Signatures”, tell-tale byte structures that begin at the start of a file. In some cases this is just a header, and in some cases there is also a footer.



War Story – A New Business in Unallocated

Timeline Analysis

- There is a wealth of Time/Date stamps stored on modern computers.
- Timelining is an analysis technique that attempts to collect all timestamps, collate them into a single dataset, and try to create sense of the activity that was occurring as specific times.
- Its is extremely helpful in trying to understand what activities occurred at critical times during an investigation.
- We will look at filesystem time initially, but as we add additional artefacts over the next few weeks, think about how you could add them into the timeline to draw a richer picture.

FAT32 Dates

FAT32 Directory Entries Store the following dates:

- Create Date & Time
- Modify Date and Time
- Access Date **ONLY** (no time)

When access date is viewed through a Windows OS, the time is reflected as 12:00:00 AM (midnight).

Its important to note that the time is stored in the local time zone of the operating system, and do not contain any timezone flags.

What implications does this have considering FAT32 likely use on removable media?

FAT32 - Directory Entry
Filename
Extension
Attributes
Reserved
Create Time
Create Date
Last Access Date
First Cluster (Most Sig. Bits)
Last Mod Time
Last Mod Date
First Cluster (Least Sig. Bits)
File Size

NTFS Dates

NTFS Time / Datestamps are stored in the $Standard_Informatrion attribute of each $MFT File record.

Timestamps available are:

- Created Date and Time
- Last Modified Date and Time
- Last Accessed Date and Time
- Changed Date and Time

Importantly, NTFS dates are stored in UTC. Assuming the date was correctly set on the source system, we can apply a time zone offset from UTC to know the local time.

We refer to these and by the acronym MACB times. Modify, Access, Change, and Birth. This is to distinguish between Change and Create.

Update Rules - $Standard_Information

SANS COMPUTER FORENSICS and INCIDENT RESPONSE

File Rename	Local File Move	Volume File Move	File Copy	File Access	File Modify	File Creation	File Deletion
Modified – No Change	Modified – No Change	Modified – No Change	Modified – No Change	Modified – No Change	Modified – Change	Modified – Change	Modified – No Change
Access – No Change	Access – No Change	Access – Changed	Access - Changed	Access – Changed (No Change on Vista/Win7)	Access – No Change	Access – Change	Access – No Change
Creation - No Change	Creation - No Change	Creation - No Change	Creation – Changed	Creation – No Change	Creation – No Change	Creation – Change	Creation - No Change
Metadata – Changed	Metadata – Changed	Metadata – Changed	Metadata - Changed	Metadata – No Change	Metadata – No Change	Metadata – Change	Metadata - No Change

What can we tell from just times?

Modify	01/01/2015 15:23
Access	02/01/2015 09:10
Change	02/01/2015 09:10
Birth	02/01/2015 09:10

We can tell this is a copy of another file, because the create time is after the modification time.

Modify	10/01/2015 12:30
Access	10/01/2015 12:30
Change	22/01/2015 22:47
Birth	10/01/2015 12:30

We can tell this file has either been renamed or moved after it was created, as it’s the only way the Change timestamp can update without any other fields updating.

Timeline Analysis

Armed with this information, there is a lot you can tell just by ordering each file by their timestamps.

There are some great tools that can extract and create a full disk timeline. (Look up fls + mactime commands for kali)

The log2timeline tool is one of the most popular ways to create a “super timeline” that can include other time information from event logs, the registry and many other sources.

Example Analysis 1

Tue Mar 24 2015 06:56:10	82	macb	r/rrwxrwxrwx	0	0	72135-48-2	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_00012f ($FILE_NAME)
Tue Mar 24 2015 06:56:10	202307	macb	r/rrwxrwxrwx	0	0	72136-128-3	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000130
Tue Mar 24 2015 06:56:10	82	macb	r/rrwxrwxrwx	0	0	72136-48-2	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000130 ($FILE_NAME)
Tue Mar 24 2015 06:56:15	117921	macb	r/rrwxrwxrwx	0	0	72137-128-3	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000132
Tue Mar 24 2015 06:56:15	82	macb	r/rrwxrwxrwx	0	0	72137-48-2	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000132 ($FILE_NAME)
Tue Mar 24 2015 06:56:20	738149	macb	r/rrwxrwxrwx	0	0	72127-128-3	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000133
Tue Mar 24 2015 06:56:20	82	macb	r/rrwxrwxrwx	0	0	72127-48-2	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000133 ($FILE_NAME)
Tue Mar 24 2015 06:56:20	21108	macb	r/rrwxrwxrwx	0	0	72139-128-3	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000134
Tue Mar 24 2015 06:56:20	82	macb	r/rrwxrwxrwx	0	0	72139-48-2	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000134 ($FILE_NAME)
Tue Mar 24 2015 06:56:20	23040	macb	r/rrwxrwxrwx	0	0	72140-128-3	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000135
Tue Mar 24 2015 06:56:20	82	macb	r/rrwxrwxrwx	0	0	72140-48-2	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000135 ($FILE_NAME)
Tue Mar 24 2015 06:56:21	20560	macb	r/rrwxrwxrwx	0	0	72141-128-3	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000136
Tue Mar 24 2015 06:56:21	82	macb	r/rrwxrwxrwx	0	0	72141-48-2	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000136 ($FILE_NAME)
Tue Mar 24 2015 06:56:21	18652	macb	r/rrwxrwxrwx	0	0	72142-128-3	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000137
Tue Mar 24 2015 06:56:21	82	macb	r/rrwxrwxrwx	0	0	72142-48-2	/Users/informant/AppData/Local/Google/Chrome/User Data/Default/Cache/f_000137 ($FILE_NAME)
Tue Mar 24 2015 06:56:30	880208	.a.b	r/rrwxrwxrwx	0	0	72145-128-5	/Users/informant/Downloads/googledrivesync.exe
Tue Mar 24 2015 06:56:30	26	.a.b	r/rrwxrwxrwx	0	0	72145-128-8	/Users/informant/Downloads/googledrivesync.exe:Zone.Identifier
Tue Mar 24 2015 06:56:30	104	.a.b	r/rrwxrwxrwx	0	0	72145-48-6	/Users/informant/Downloads/googledrivesync.exe ($FILE_NAME)
Tue Mar 24 2015 06:56:31	104	m.c.	r/rrwxrwxrwx	0	0	72145-48-6	/Users/informant/Downloads/googledrivesync.exe ($FILE_NAME)
Tue Mar 24 2015 06:56:33	880208	m.c.	r/rrwxrwxrwx	0	0	72145-128-5	/Users/informant/Downloads/googledrivesync.exe
Tue Mar 24 2015 06:56:33	26	m.c.	r/rrwxrwxrwx	0	0	72145-128-8	/Users/informant/Downloads/googledrivesync.exe:Zone.Identifier
Tue Mar 24 2015 06:56:52	96	m.c.	r/rrwxrwxrwx	0	0	72096-48-6	/Users/informant/Downloads/icloudsetup.exe ($FILE_NAME)
Tue Mar 24 2015 06:56:53	56	mac.	d/d-wx-wx-wx	0	0	527-144-7	/Users/informant/Downloads
Tue Mar 24 2015 06:56:53	71647536	m.c.	r/rrwxrwxrwx	0	0	72096-128-5	/Users/informant/Downloads/icloudsetup.exe
Tue Mar 24 2015 06:56:53	26	m.c.	r/rrwxrwxrwx	0	0	72096-128-8	/Users/informant/Downloads/icloudsetup.exe:Zone.Identifier
Tue Mar 24 2015 07:00:16	416	macb	r/rrwxrwxrwx	0	0	71736-128-1	/Users/informant/AppData/LocalLow/Microsoft/CryptnetUrlCache/MetaData/7B8944BA8AD0EFDF0E01A43EF62BECD0_
Tue Mar 24 2015 07:00:16	196	macb	r/rrwxrwxrwx	0	0	71736-48-2	/Users/informant/AppData/LocalLow/Microsoft/CryptnetUrlCache/MetaData/7B8944BA8AD0EFDF0E01A43EF62BECD0_
Tue Mar 24 2015 07:00:16	1725	macb	r/rrwxrwxrwx	0	0	72147-128-4	/Users/informant/AppData/LocalLow/Microsoft/CryptnetUrlCache/Content/7B8944BA8AD0EFDF0E01A43EF62BECD0_
Tue Mar 24 2015 07:00:16	196	macb	r/rrwxrwxrwx	0	0	72147-48-2	/Users/informant/AppData/LocalLow/Microsoft/CryptnetUrlCache/Content/7B8944BA8AD0EFDF0E01A43EF62BECD0_
Tue Mar 24 2015 07:00:18	2481	.a.b	r/rrwxrwxrwx	0	0	72149-128-4	/Users/informant/AppData/Local/Temp/icloudsetupE88.log
Tue Mar 24 2015 07:00:18	102	macb	r/rrwxrwxrwx	0	0	72149-48-2	/Users/informant/AppData/Local/Temp/icloudsetupE88.log ($FILE_NAME)
Tue Mar 24 2015 07:00:31	1032341	.a.b	r/rrwxrwxrwx	0	0	72158-128-4	/Users/informant/AppData/Local/Temp/SetupAdmin394.log
Tue Mar 24 2015 07:00:31	100	macb	r/rrwxrwxrwx	0	0	72158-48-2	/Users/informant/AppData/Local/Temp/SetupAdmin394.log ($FILE_NAME)

Example Analysis 2

Date	Size	Type	Mode	Meta	File Name
Thu Mar 26 2015 01:47:52	28967	macb	r/rrwxrwxrwx	75119-128-4	/Users/informant/AppData/Local/Microsoft/Windows/Temporary Internet Files/Low/Content.IE5/BU7W71VG/ccleaner[1].htm
Thu Mar 26 2015 01:47:52	96	macb	r/rrwxrwxrwx	75119-48-2	/Users/informant/AppData/Local/Microsoft/Windows/Temporary Internet Files/Low/Content.IE5/BU7W71VG/ccleaner[1].htm ($FILE_NAME)
Thu Mar 26 2015 01:47:55	851	macb	r/rrwxrwxrwx	75162-128-4	/Users/informant/AppData/Local/Microsoft/Windows/Temporary Internet Files/Low/Content.IE5/OOYOT1KJ/ccleaner[1].png
Thu Mar 26 2015 01:47:55	96	macb	r/rrwxrwxrwx	75162-48-2	/Users/informant/AppData/Local/Microsoft/Windows/Temporary Internet Files/Low/Content.IE5/OOYOT1KJ/ccleaner[1].png ($FILE_NAME)
Thu Mar 26 2015 01:58:34	48	...b	-/drwxrwxrwx	75246-144-1	/Program Files/CCleaner (deleted)
Thu Mar 26 2015 01:58:34	82	macb	-/drwxrwxrwx	75246-48-2	/Program Files/CCleaner ($FILE_NAME) (deleted)
Thu Mar 26 2015 01:58:35	5529880	.a..	-/rrwxrwxrwx	75248-128-3	/Program Files/CCleaner/CCleaner.exe (deleted)
Thu Mar 26 2015 01:58:35	90	macb	-/rrwxrwxrwx	75248-48-2	/Program Files/CCleaner/CCleaner.exe ($FILE_NAME) (deleted)
Thu Mar 26 2015 01:58:35	7451928	.a..	-/rrwxrwxrwx	75250-128-4	/Program Files/CCleaner/CCleaner64.exe (deleted)
Thu Mar 26 2015 01:58:35	94	macb	-/rrwxrwxrwx	75250-48-2	/Program Files/CCleaner/CCleaner64.exe ($FILE_NAME) (deleted)
Thu Mar 26 2015 01:58:35	822	ma.b	-/rrwxrwxrwx	75306-128-3	/Users/Public/Desktop/CCleaner.lnk (deleted)
Thu Mar 26 2015 01:58:35	90	macb	-/rrwxrwxrwx	75306-48-2	/Users/Public/Desktop/CCleaner.lnk ($FILE_NAME) (deleted)
Thu Mar 26 2015 01:58:35	154384	.a..	-/rrwxrwxrwx	75307-128-3	/Program Files/CCleaner/uninst.exe (deleted)
Thu Mar 26 2015 01:58:35	86	macb	-/rrwxrwxrwx	75307-48-2	/Program Files/CCleaner/uninst.exe ($FILE_NAME) (deleted)
Thu Mar 26 2015 01:58:37	29954	.a.b	r/rrwxrwxrwx	75309-128-4	/Windows/Prefetch/CCLEANER64.EXE-779BD542.pf
Thu Mar 26 2015 01:58:37	118	macb	r/rrwxrwxrwx	75309-48-2	/Windows/Prefetch/CCLEANER64.EXE-779BD542.pf ($FILE_NAME)
Thu Mar 26 2015 02:15:50	29954	m.c.	r/rrwxrwxrwx	75309-128-4	/Windows/Prefetch/CCLEANER64.EXE-779BD542.pf
Thu Mar 26 2015 02:18:36	5529880	..c.	-/rrwxrwxrwx	75248-128-3	/Program Files/CCleaner/CCleaner.exe (deleted)
Thu Mar 26 2015 02:18:36	7451928	..c.	-/rrwxrwxrwx	75250-128-4	/Program Files/CCleaner/CCleaner64.exe (deleted)
Thu Mar 26 2015 02:18:36	154384	..c.	-/rrwxrwxrwx	75307-128-3	/Program Files/CCleaner/uninst.exe (deleted)
Thu Mar 26 2015 02:18:37	48	mac.	-/drwxrwxrwx	75246-144-1	/Program Files/CCleaner (deleted)
Thu Mar 26 2015 02:18:37	822	..c.	-/rrwxrwxrwx	75306-128-3	/Users/Public/Desktop/CCleaner.lnk (deleted)

NTFS Dates - $Filename

- The $Filename attribute also has its own set of MACB times, separate to the $Standard_Information Times

- This is important, as the $Standard_Information MACB times can be vulnerable to an anti-forensics technique known as timestomping.

- The $Filename timestamps are much more difficult to manipulate directly, so comparisons can potentially reveal traces of **timestomping**.

- There are different rules for $Filename times.

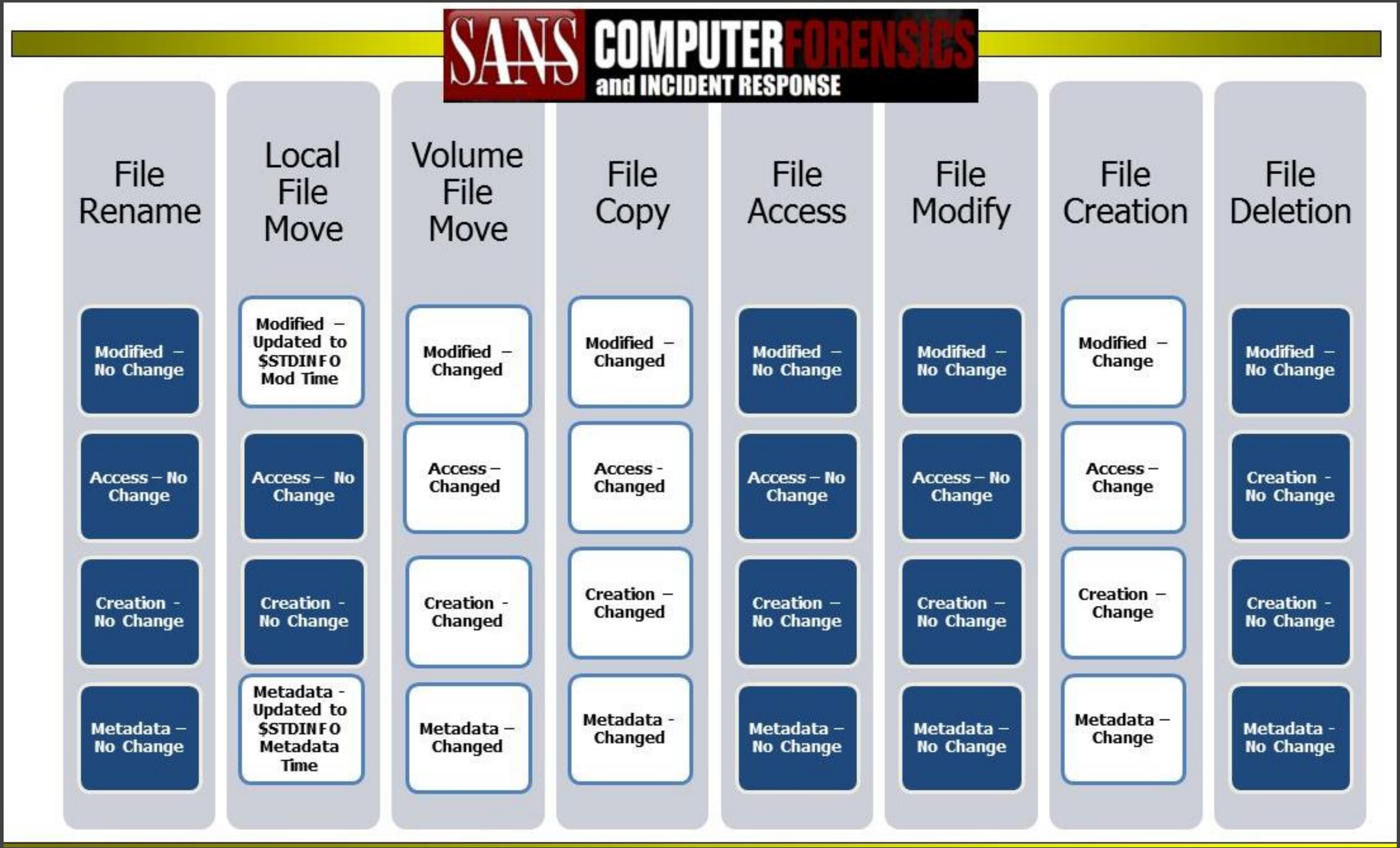
```
$ istat -o 206848 cfreds_2015_data_leakage_pc.E01 72606

MFT Entry Header Values:
Entry: 72606        Sequence: 1
$LogFile Sequence Number: 320667379
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Archive
Owner ID: 0
Security ID: 452  (S-1-5-18)
Last User Journal Update Sequence Number: 56643936
Created:  2014-07-31 16:16:24.000000000 (UTC)
File Modified:      2014-07-31 16:16:24.000000000 (UTC)
MFT Modified:       2015-03-23 20:00:43.047074400 (UTC)
Accessed: 2015-03-23 20:00:43.047074400 (UTC)

$FILE_NAME Attribute Values:
Flags: Archive
Name: LICENSE.txt
Parent MFT Entry: 72603        Sequence: 1
Allocated Size: 0    Actual Size: 0
Created:  2015-03-23 20:00:43.031474400 (UTC)
File Modified:      2015-03-23 20:00:43.031474400 (UTC)
MFT Modified:       2015-03-23 20:00:43.031474400 (UTC)
Accessed: 2015-03-23 20:00:43.031474400 (UTC)
```

Update Rules - $Filename



SANS COMPUTER FORENSICS and INCIDENT RESPONSE

File Rename	Local File Move	Volume File Move	File Copy	File Access	File Modify	File Creation	File Deletion
Modified – No Change	Modified – Updated to $STDINFO Mod Time	Modified – Changed	Modified – Changed	Modified – No Change	Modified – No Change	Modified – Change	Modified – No Change
Access – No Change	Access – No Change	Access – Changed	Access - Changed	Access – No Change	Access – No Change	Access – Change	Creation - No Change
Creation - No Change	Creation - No Change	Creation - Changed	Creation – Changed	Creation – No Change	Creation – No Change	Creation – Change	Creation - No Change
Metadata – No Change	Metadata - Updated to $STDINFO Metadata Time	Metadata – Changed	Metadata - Changed	Metadata – No Change	Metadata – No Change	Metadata – Change	Metadata - No Change

Anti-forensics - Timestomping

Timestomping is the attempt to hide your tracks by artificially changing the timestamps of files in the MFT.

Scenario:

An adversary breaches your environment. Your network logs indicate the breach commenced on 9PM Friday night, 24 hours ago. You commence investigating files created or modifies in the past 24 hours.

Problem:

They have also timestomped their malware to look as though they were created, accessed and modified when windows was initially installed 3 years ago. Your analysis misses these critical files.

```
C:\>timestomp text.txt -m "Monday 1/01/2001 01:01:1 AM"

C:\>dir
 Volume in drive C has no label.
 Volume Serial Number is 3036-18D7

 Directory of C:\

05/26/2007  06:01 PM                 0 AUTOEXEC.BAT
05/26/2007  06:01 PM                 0 CONFIG.SYS
10/24/2007  02:58 PM    <DIR>          Documents and Settings
05/29/2007  01:15 PM    <DIR>          Program Files
01/01/2001  01:01 AM                 0 text.txt
10/24/2007  02:50 PM            57,344 timestomp.exe
06/18/2007  05:31 PM    <DIR>          WINDOWS
               4 File(s)         57,344 bytes
               3 Dir(s)  11,767,320,576 bytes free

C:\>timestomp text.txt -v
Modified:                  Monday 1/1/2001 1:1:1
Accessed:               Wednesday 10/24/2007 14:51:2
Created:                Wednesday 10/24/2007 14:51:2
Entry Modified:         Wednesday 10/24/2007 14:59:8

C:\>_
```

Detection

Comparing the $Standard_information and $Filename timestamps can reveal timestomping.

$Standard_Information	
Modify	01/01/2015 15:23
Access	02/01/2015 09:10
Change	02/01/2015 09:10
Birth	02/01/2015 09:10

$Filename	
Modify	31/07/2018 12:05
Access	31/07/2018 12:05
Change	31/07/2018 12:05
Birth	31/07/2018 12:05

A significant difference in the timestamps is evidence timestomping may have occurred.

War Story - The Stolen Laptop