RECONSTRUCTING THE MBR AND MFT FROM MEMORY

Jamie “Gleeda” Levy

OMFW 2012

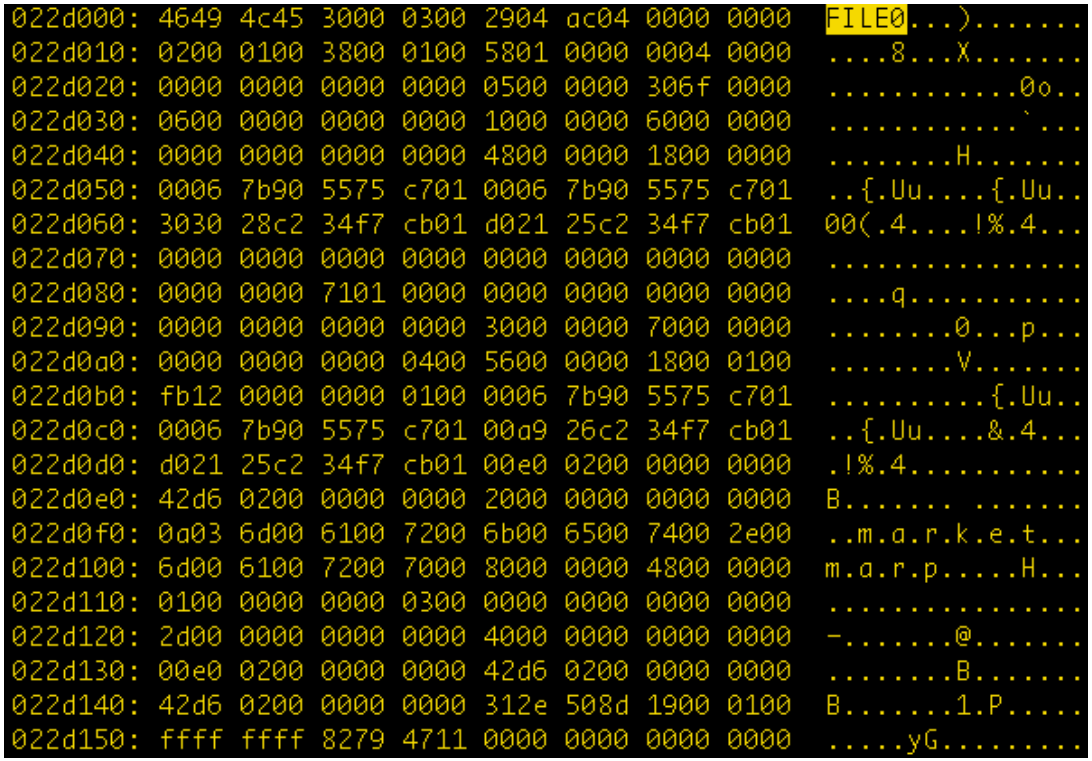
Disk Artifacts

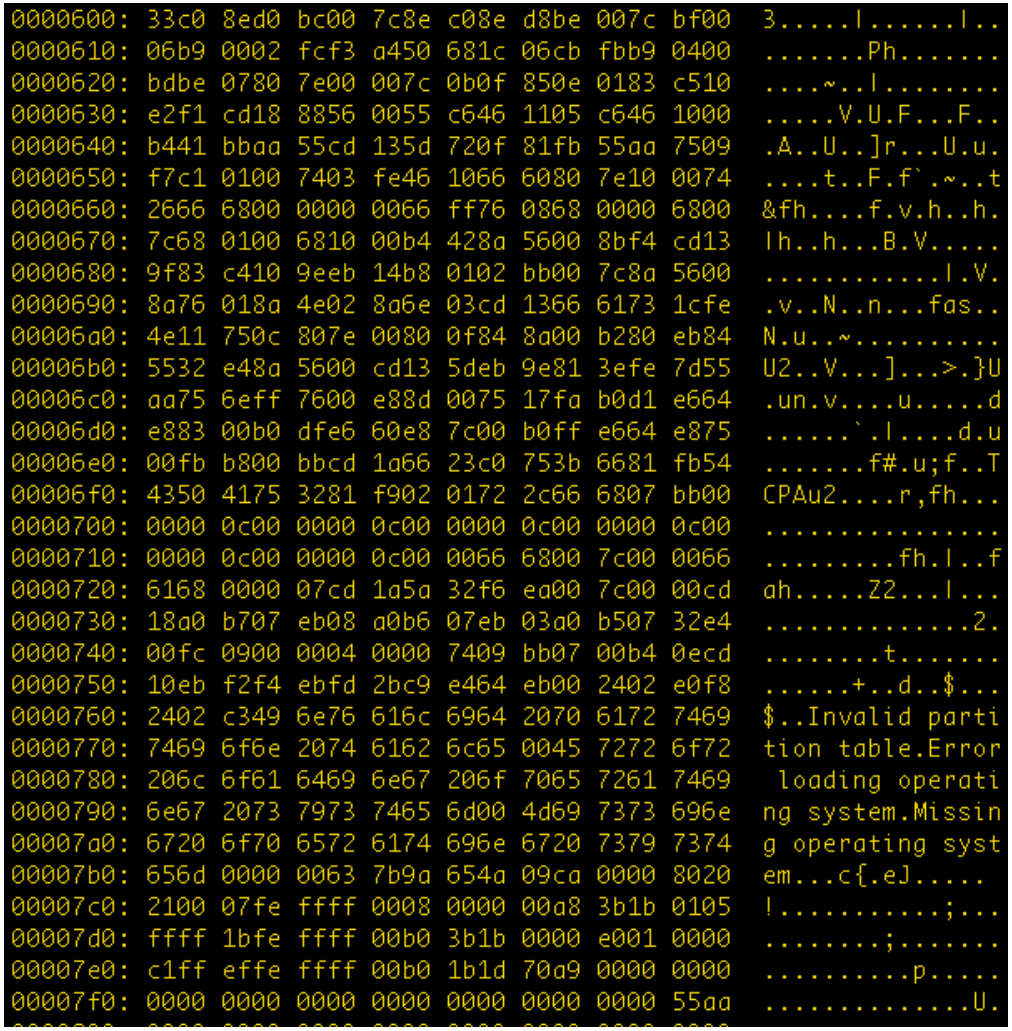
- Disk artifacts can be found in memory
- These can be useful in an investigation

- MBR
  - Able to scan for changed MBRs or particular MBR bootcode
- MFT Entries
  - Able to find file names
  - MAC times
  - Paths

In the beginning….

- Able to find MBR and MFT entries in memory by using `strings` or a hex editor
- Able to parse out by hand




0000600: 33c0 8ed0 bc00 7c8e c08e d8be 007c bf00 3.....|......|..
0000610: 06b9 0002 fcf3 a450 681c 06cb fbb9 0400 ....Ph.......
0000620: bdbe 0780 7e00 007c 0b0f 850e 0183 c510 ....~..|........
0000630: e2f1 cd18 8856 0055 c646 1105 c646 1000 .....V.U.F...F..
0000640: b441 bbaa 55cd 135d 720f 81fb 55aa 7509 .A..U..]r...U.u.
0000650: f7c1 0100 7403 fe46 1066 6080 7e10 0074 ....t..F.f`.~..t
0000660: 2666 6800 0000 0066 ff76 0868 0000 6800 &fh....f.v.h..h.
0000670: 7c68 0100 6810 00b4 428a 5600 8bf4 cd13 |h..h...B.V.....
0000680: 9f83 c410 9eeb 14b8 0102 bb00 7c8a 5600 ....|.V.
0000690: 8a76 018a 4e02 8a6e 03cd 1366 6173 1cfe .v..N..n...fas..
00006a0: 4e11 750c 807e 0080 0f84 8a00 b280 eb84 N.u..~..........
00006b0: 5532 e48a 5600 cd13 5deb 9e81 3efe 7d55 U2..V...]...>.}U
00006c0: aa75 6eff 7600 e88d 0075 17fa b0d1 e664 .un.v....u.....d
00006d0: e883 00b0 dfe6 60e8 7c00 b0ff e664 e875 ......`.|....d.u
00006e0: 00fb b800 bbcd 1a66 23c0 753b 6681 fb54 ....f#.u;f..T
00006f0: 4350 4175 3281 f902 0172 2c66 6807 bb00 CPAu2....r,fh...
0000700: 0000 0c00 0000 0c00 0000 0c00 0000 0c00 ....
0000710: 0000 0c00 0000 0c00 0066 6800 7c00 0066 ....fh.|..f
0000720: 6168 0000 07cd 1a5a 32f6 ea00 7c00 00cd ah.....Z2...|...
0000730: 18a0 b707 eb08 a0b6 07eb 03a0 b507 32e4 ....2.
0000740: 00fc 0900 0004 0000 7409 bb07 00b4 0ecd ....t.......
0000750: 10eb f2f4 ebfd 2bc9 e464 eb00 2402 e0f8 ......+..d..$...
0000760: 2402 c349 6e76 616c 6964 2070 6172 7469 $..Invalid parti
0000770: 7469 6f6e 2074 6162 6c65 0045 7272 6f72 tion table.Error
0000780: 206c 6f61 6469 6e67 206f 7065 7261 7469   loading operati
0000790: 6e67 2073 7973 7465 6d00 4d69 7373 696e ng system.Missin
00007a0: 6720 6f70 6572 6174 696e 6720 7379 7374 g operating syst
00007b0: 656d 0000 0063 7b9a 654a 09ca 0000 8020 em...c{.eJ.....
00007c0: 2100 07fe ffff 0008 0000 00a8 3b1b 0105 !...........;...
00007d0: ffff 1bfe ffff 00b0 3b1b 0000 e001 0000 ....;.......
00007e0: c1ff effe ffff 00b0 1b1d 70a9 0000 0000 ....p.....
00007f0: 0000 0000 0000 0000 0000 0000 0000 55aa ....U.

Finding disk artifacts

- First we need to know what these artifacts “look like”
  - What’s common?
  - Patterns
- Second we need to have a more intelligent way to represent these artifacts
  - Structures
  - Parse out information
  - Present in a clear way

MASTER BOOT RECORD

Master Boot Record (MBR)

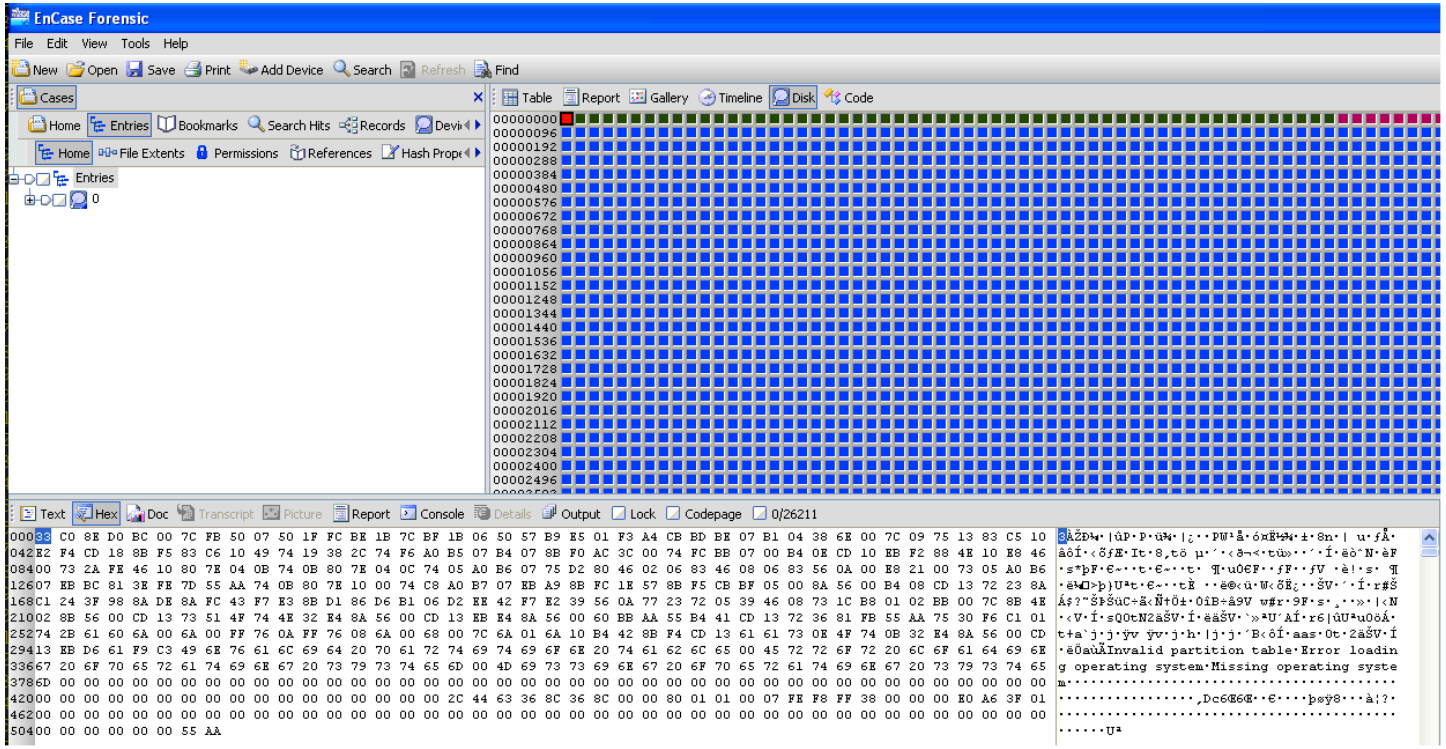
- Why?

- Used to boot the machine after POST operations
- Holds information about partitions

- A known infection vector for malware

MBR

- Located within the first 512 bytes of the disk



MBR Structure – Boot code

```
0000000: 33c0 8ed0 bc00 7cfb 5007 501f fcbe 1b7c  3.....|.P.P....|
0000010: bf1b 0650 57b9 e501 f3a4 cbbd be07 b104  ...PW...........
0000020: 386e 007c 0975 1383 c510 e2f4 cd18 8bf5  8n.|.u..........
0000030: 83c6 1049 7419 382c 74f6 a0b5 07b4 078b  ...It.8,t.......
0000040: f0ac 3c00 74fc bb07 00b4 0ecd 10eb f288  ..<.t...........
0000050: 4e10 e846 0073 2afe 4610 807e 040b 740b  N..F.s*.F..~..t.
0000060: 807e 040c 7405 a0b6 0775 d280 4602 0683  .~..t....u..F...
0000070: 4608 0683 560a 00e8 2100 7305 a0b6 07eb  F...V...!.s.....
0000080: bc81 3efe 7d55 aa74 0b80 7e10 0074 c8a0  ..>.}U.t..~..t..
0000090: b707 eba9 8bfc 1e57 8bf5 cbbf 0500 8a56  .......W.......V
00000a0: 00b4 08cd 1372 238a c124 3f98 8ade 8afc  .....r#..$?.....
00000b0: 43f7 e38b d186 d6b1 06d2 ee42 f7e2 3956  C..........B..9V
00000c0: 0a77 2372 0539 4608 731c b801 02bb 007c  .w#r.9F.s......|
00000d0: 8b4e 028b 5600 cd13 7351 4f74 4e32 e48a  .N..V...sQOtN2..
00000e0: 5600 cd13 ebe4 8a56 0060 bbaa 55b4 41cd  V......V.`..U.A.
00000f0: 1372 3681 fb55 aa75 30f6 c101 742b 6160  .r6..U.u0...t+a`
0000100: 6a00 6a00 ff76 0aff 7608 6a00 6800 7c6a  j.j..v..v.j.h.|j
0000110: 016a 10b4 428b f4cd 1361 6173 0e4f 740b  .j..B....aas.Ot.
0000120: 32e4 8a56 00cd 13eb d661 f9c3 496e 7661  2..V.....a..
```

MBR Structure – Error Messages

```
0000120:                                     496e 7661                Inva
0000130: 6c69 6420 7061 7274 6974 696f 6e20 7461  lid partition ta
0000140: 626c 6500 4572 726f 7220 6c6f 6164 696e  ble.Error loadin
0000150: 6720 6f70 6572 6174 696e 6720 7379 7374  g operating syst
0000160: 656d 004d 6973 7369 6e67 206f 7065 7261  em.Missing opera
0000170: 7469 6e67 2073 7973 7465 6d00 0000 0000  ting system.....
```

MBR Structure – Partition Table

```
00001b0:                                         8001                ..
00001c0: 0100 07fe f8ff 3800 0000 e0a6 3f01 0000  ......8.....?...
00001d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00001e0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00001f0: 0000 0000 0000 0000 0000 0000 0000 55aa  ..............U.
```

MBR Structure

```
>>> dt("PARTITION_ENTRY")
'PARTITION_ENTRY' (16 bytes)
0x0   : BootableFlag                    ['char']
0x1   : StartingCHS                     ['array', 3, ['unsigned char']]
0x4   : PartitionType                   ['char']
0x5   : EndingCHS                       ['array', 3, ['unsigned char']]
0x8   : StartingLBA                     ['unsigned int']
0xc   : SizeInSectors                   ['int']

>>> dt("PARTITION_TABLE")
'PARTITION_TABLE' (512 bytes)
0x1b8 : DiskSignature                   ['array', 4, ['unsigned char']]
0x1bc : Unused                          ['unsigned short']
0x1be : Entry1                          ['PARTITION_ENTRY']
0x1ce : Entry2                          ['PARTITION_ENTRY']
0x1de : Entry3                          ['PARTITION_ENTRY']
0x1ee : Entry4                          ['PARTITION_ENTRY']
0x1fe : Signature                       ['unsigned short']
```

MBR Structure – Boot code

- Usually the first 440 bytes of the MBR
- Actual code can be less than this

- Contains instructions for loading its own code into memory
- Finding active bootable partitions
- Load the active bootable partition's Volume Boot Record (VBR)

MBR Structure – Boot code

```
0x00000000: 33c0          XOR AX, AX
0x00000002: 8ed0          MOV SS, AX
0x00000004: bc007c        MOV SP, 0x7c00
0x00000007: fb            STI
0x00000008: 50            PUSH AX
0x00000009: 07            POP ES
0x0000000a: 50            PUSH AX
0x0000000b: 1f            POP DS
0x0000000c: fc            CLD
0x0000000d: be1b7c        MOV SI, 0x7c1b
0x00000010: bf1b06        MOV DI, 0x61b
0x00000013: 50            PUSH AX
0x00000014: 57            PUSH DI
0x00000015: b9e501        MOV CX, 0x1e5
0x00000018: f3a4          REP MOVSB
0x0000001a: cb            RETF
0x0000001b: bdbe07        MOV BP, 0x7be
[snip]
```

MBR Structure – Boot code

- Boot code is the same for windows machines of the same architecture
- For example, all Windows XP x86 machines should have the same boot code
- Makes it easy to check on disk
- Memory is slightly different

XP MBR in memory [1]

- After the computer executes POST the BIOS loads the MBR into physical memory at 0x7c00 and transfers control to this code
- The code must copy itself into another part of memory
  - Overwrites current location with active partition
  - It doesn’t copy over already executed code, but writes from code offset 0x1b-0x1ff to physical memory offsets 0x61b-0x7ff

XP MBR in memory – 0x600

```
0x00000000: 0000          ADD [BX+SI], AL
0x00000002: 0000          ADD [BX+SI], AL
0x00000004: 0000          ADD [BX+SI], AL
0x00000006: 0000          ADD [BX+SI], AL
0x00000008: 0000          ADD [BX+SI], AL
0x0000000a: 0000          ADD [BX+SI], AL
0x0000000c: 0000          ADD [BX+SI], AL
0x0000000e: 0000          ADD [BX+SI], AL
0x00000010: 0000          ADD [BX+SI], AL
0x00000012: 0000          ADD [BX+SI], AL
0x00000014: 0000          ADD [BX+SI], AL
0x00000016: 0000          ADD [BX+SI], AL
0x00000018: 0000          ADD [BX+SI], AL
0x0000001a: 00bdbe07      ADD [DI+0x7be], BH
0x0000001e: b104          MOV CL, 0x4  ;ok from here
0x00000020: 386e00        CMP [BP+0x0], CH
0x00000023: 7c09          JL 0x2e
```

XP MBR in memory – 0x61b

```
0x00000000: bdbe07        MOV BP, 0x7be
0x00000003: b104          MOV CL, 0x4
0x00000005: 386e00        CMP [BP+0x0], CH
0x00000008: 7c09          JL 0x13
0x0000000a: 7513          JNZ 0x1f
0x0000000c: 83c510        ADD BP, 0x10
0x0000000f: e2f4          LOOP 0x5
0x00000011: cd18          INT 0x18
0x00000013: 8bf5          MOV SI, BP
0x00000015: 83c610        ADD SI, 0x10
0x00000018: 49            DEC CX
0x00000019: 7419          JZ 0x34
0x0000001b: 382c          CMP [SI], CH
0x0000001d: 74f6          JZ 0x15
```

Vista+ MBR in memory [2-3]

- After the computer executes POST the BIOS loads the MBR into physical memory at 0x7c00 and transfers control to this code
- The code must copy itself into another part of memory
  - Overwrites current location with active partition
  - Copies entire 512 bytes to physical memory offsets 0x600-0x7ff
- This should be easy right?

Vista+ MBR in memory – Wrong!

```
0x000000fb: 666807bb0000    PUSH DWORD 0xbb07
0x00000101: 000c            ADD [SI], CL
0x00000103: 0000            ADD [BX+SI], AL
0x00000105: 000c            ADD [SI], CL
0x00000107: 0000            ADD [BX+SI], AL
0x00000109: 000c            ADD [SI], CL
0x0000010b: 0000            ADD [BX+SI], AL
0x0000010d: 000c            ADD [SI], CL
0x0000010f: 0000            ADD [BX+SI], AL
0x00000111: 000c            ADD [SI], CL
0x00000113: 0000            ADD [BX+SI], AL
0x00000115: 000c            ADD [SI], CL
0x00000117: 0000            ADD [BX+SI], AL
0x00000119: 6668007c0000    PUSH DWORD 0x7c00
0x0000011f: 6661            POPA
0x00000121: 680000          PUSH WORD 0x0
0x00000124: 07              POP ES
0x00000125: cd1a            INT 0x1a
0x00000127: 5a              POP DX
```

```
0x000000fb: 666807bb0000    PUSH DWORD 0xbb07
0x00000101: 666800020000    PUSH DWORD 0x200
0x00000107: 666808000000    PUSH DWORD 0x8
0x0000010d: 6653            PUSH EBX
0x0000010f: 6653            PUSH EBX
0x00000111: 6655            PUSH EBP
0x00000113: 666800000000    PUSH DWORD 0x0
0x00000119: 6668007c0000    PUSH DWORD 0x7c00
0x0000011f: 6661            POPA
0x00000121: 680000          PUSH WORD 0x0
0x00000124: 07              POP ES
0x00000125: cd1a            INT 0x1a
0x00000127: 5a              POP DX
```

Methodology

- Check MBR at 0x600 or 0x61b
- How?
  - Carve out MBR starting at offset
- Pros?
  - Fast, often able to find discrepancies
- Issues?
  - Malware may copy the original MBR code to the proper location
  - May miss malicious MBR

Methodology

- Scan for all potential MBRs
- How?
  - Scan for ‘\x55\xaa’
- Pros?
  - Less potential to miss
- Issues?
  - May get a lot of false positives

Methodology

- Scan for all potential MBRs but discard those that are possibly invalid

- How?
  - Scan for ‘\x55\xaa’
  - Check partition table

- Pros?
  - Less potential to miss
  - Less false positives

Methodology

- Naïve scan for an MBR with a particular boot code hash
- How?
  - Hash first 440 bytes
- Pros?
  - Yields less false positives
- Issues?
  - Boot code may differ depending on message strings which are inconsequential

Methodology

- Scan for an MBR with a particular boot code hash but only look at executable code

- How?
  - Use the disassembly to find the RET instruction
  - Hash the data before the RET

- Pros?
  - Scan is not effected by message strings

Example: tdl4

- In experiments a Windows XP x86 VM was infected with a tdl4 variant
- VM was restarted and suspended
- MBR was extracted from disk

- Memory was examined from offset 0x600

Example: tdl4 – offset 0x600

```
0x00000000: 33c0          XOR AX, AX
0x00000002: 8ed0          MOV SS, AX
0x00000004: bc007c        MOV SP, 0x7c00
0x00000007: 8ec0          MOV ES, AX
0x00000009: 8ed8          MOV DS, AX
0x0000000b: be007c        MOV SI, 0x7c00
0x0000000e: bf0006        MOV DI, 0x600
0x00000011: b90002        MOV CX, 0x200
0x00000014: fc            CLD
0x00000015: f3a4          REP MOVSB
0x00000017: 50            PUSH AX
0x00000018: 681c06        PUSH WORD 0x61c
0x0000001b: bdbe07        MOV BP, 0x7be
0x0000001e: b104          MOV CL, 0x4
```

Example: tdl4

- We see that something is amiss…
- We could scan for all MBRs and see if there are others that look suspicious to find the malicious one

Example: tdl4

- If the malicious MBR hash is known ahead of time, we can search for its hash:

- Here memory is searched using the hash of the boot code up to the RET instruction of the MBR from disk:

```
$ python vol.py -f tdl4.vmem mbrparser -M
5429921523056d910bf95f32da2b2ab4
```

Example: tdl4

```
Potential MBR at physical offset: 0x60f00
Disk Signature: 19-df-19-df
Bootcode md5: 5429921523056d910bf95f32da2b2ab4
Bootcode (FULL) md5: 2839639fa37b8353e792a2a30a12ced3
Disassembly of Bootable Code:
0x00000000: 33c0                                XOR AX, AX
0x00000002: 8ed0                                MOV SS, AX
0x00000004: bc007c                              MOV SP, 0x7c00
0x00000007: 8ec0                                MOV ES, AX
0x00000009: 8ed8                                MOV DS, AX
0x0000000b: be007c                              MOV SI, 0x7c00
0x0000000e: bf0006                              MOV DI, 0x600
0x00000011: b90002                              MOV CX, 0x200
0x00000014: fc                                  CLD
0x00000015: f3a4                                REP MOVSB
0x00000017: 50                                  PUSH AX
0x00000018: 681c06                              PUSH WORD 0x61c
0x0000001b: cb                                  RETF
0x0000001c: fb                                  STI
0x0000001d: 60                                  PUSHA
0x0000001e: b94701                              MOV CX, 0x147
0x00000021: bd2a06                              MOV BP, 0x62a
0x00000024: d24e00                              ROR [BP+0x0], CL
```

Example: tdl4

```
0x000000c6: 93                                        XCHG BX, AX
0x000000c7: 1cfd                                      SBB AL, 0xfd
0x000000c9: c3                                        RET

0x000000ca: ba 3e 51 f8 93 1c 04 04 00 32 31 7c 54 b6 e4 07   .>Q......21|T...
0x000000da: 44 eb 4e 70 44 36 e4 07 23 bf d9 57 20 a4 ad 88   D.NpD6..#..W....
0x000000ea: 6b bf 78 57 ee 22 3c 72 04 22 c3 37 40 fa 45 08   k.xW."<r.".7@.E.
0x000000fa: 5d 00 40 f0 b5 78 e4 08 87 ad c0 37 40 c4 ff fe   ].@..x.....7@...
0x0000010a: e1 a2 f1 27 38 00 91 89 e3 a2 b5 27 38 22 1b 72   ...'8......'8".r
0x0000011a: 83 22 f5 27 38 00 d3 30 f6 62 f9 a8 6c c9 0e 30   ."'8..0.b..l..0
0x0000012a: 06 91 49 57 d6 85 87 66 98 30 1d f3 ff 06 7c a2   ..IW...f.0....|.
0x0000013a: 04 d0 88 47 68 99 ff 0e 4a 02 cc 38 f0 62 10 00   ...Gh...J..8.b..
0x0000014a: f5 79 31 dc dd 40 00 be 54 02 7e 6a 49 bb 0b c9   .y1..@..T.~jI...
0x0000015a: 3a 83 20 ec 1c 1b 20 29 f4 40 f9 98 4f 2d ea ea   :..... ).@..O-..
0x0000016a: e1 1b 8c 27 89 d8 00 69 6e 67 20 73 79 73 74 65   ...'...ing.syste
0x0000017a: 6d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   m...............
0x0000018a: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x0000019a: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x000001aa: 00 00 00 00 00 00 00 00 00 00 00 2c 44 63         ...........,Dc

===== Partition Table #1 =====
Boot flag: 0x80 (Bootable)
Partition type: 0x7 (NTFS)
Starting Sector (LBA): 0x38 (56)
Starting CHS: Cylander: 0 Head: 1 Sector: 1
```

Example: Mebromi

```
$ ./vol.py –f mebromi.vmem mbrparser -M 4ad444d4e7efce9485a94186c3f4b157
Volatile Systems Volatility Framework 2.3_alpha
Potential MBR at physical offset: 0x60f00
Disk Signature: 19-df-19-df
Bootcode md5: 4ad444d4e7efce9485a94186c3f4b157
Bootcode (full) md5: 4ad444d4e7efce9485a94186c3f4b157
Disassembly of Bootable Code:
0x00000000: 33c0                                XOR AX, AX
0x00000002: 8ed0                                MOV SS, AX
0x00000004: bc007c                              MOV SP, 0x7c00
0x00000007: fb                                  STI
0x00000008: 50                                  PUSH AX
0x00000009: 07                                  POP ES
0x0000000a: 50                                  PUSH AX
0x0000000b: 1f                                  POP DS
0x0000000c: fc                                  CLD
0x0000000d: 50                                  PUSH AX
0x0000000e: be007c                              MOV SI, 0x7c00
0x00000011: bf0006                              MOV DI, 0x600
0x00000014: b90002                              MOV CX, 0x200
```

Example: Mebromi

```
0060f00: 33c0 8ed0 bc00 7cfb 5007 501f fc50 be00  3.....|.P.P..P..
0060f10: 7cbf 0006 b900 02f3 a4bf 1e06 57cb b441  |...........W..A
0060f20: b280 bbaa 55cd 1381 fb55 aa75 30f6 c101  ....U....U.u0...
0060f30: 742b be00 08c7 0410 00c7 4402 0600 c744  t+........D....D
0060f40: 0400 7cc7 4406 0000 c644 0801 b907 00bf  ..|.D....D......
0060f50: 0908 c605 0047 e2fa b800 42eb 0bb8 0602  .....G....B.....
0060f60: bb00 7cb9 0200 b600 b280 cd13 720d 80fc  ..|.........r...
0060f70: 0075 0833 c050 b800 7c50 cbb8 9206 8bf0  .u.3.P..|P......
0060f80: 8a04 3c00 740a b40e bb55 00cd 1046 ebf0  ..<.t....U...F..
0060f90: ebfe 696e 7431 3320 6572 726f 7221 0053  ..int13 error!.S
0060fa0: 7563 6365 7373 2124 0000 0000 0000 0000  uccess!$........
0060fb0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

RET ?

MASTER FILE TABLE

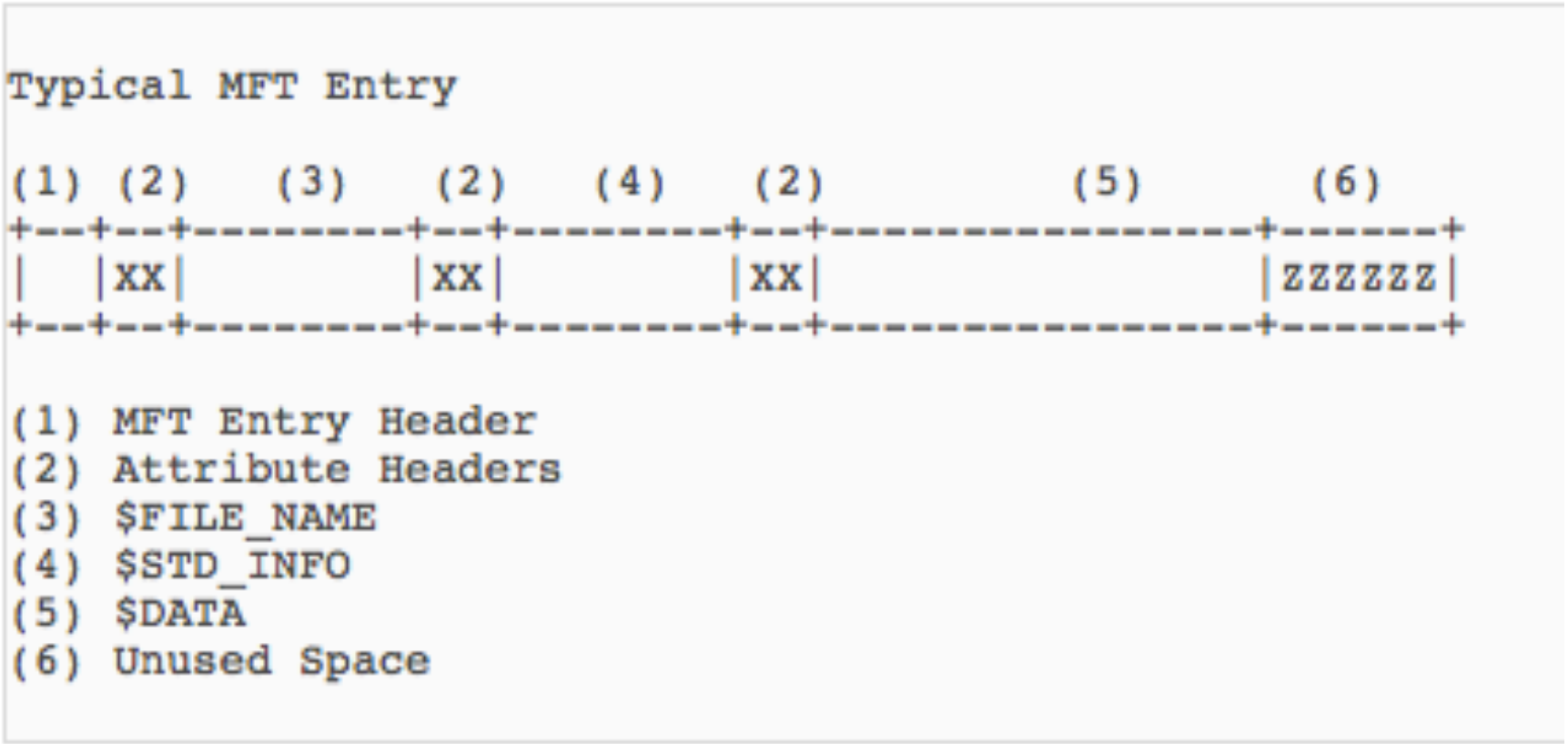
Master File Table (MFT)

- Brian Carriers book: “File System Forensic Analysis” was used heavily for this plugin

Master File Table (MFT)

- The Master File Table (MFT) store information needed to retrieve files on an NTFS file system
- It is comprised of entries
  - Each entry has a size of 1024 bytes normally, though this can be different
  - Size can be found in $Boot

Master File Table [4]



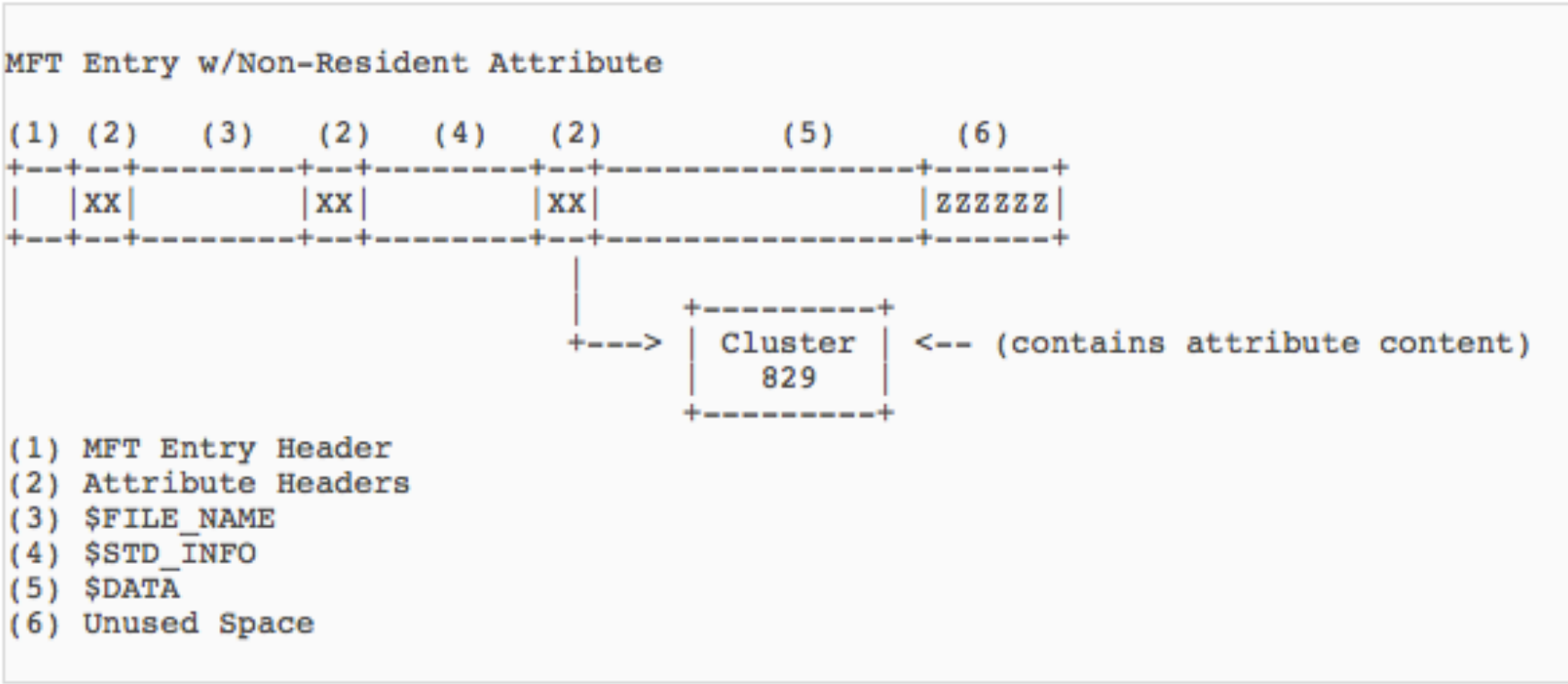
Master File Table (MFT)

- Each entry is comprised of an MFT header and attributes
- The MFT header contains various information including:
  - signature - either "FILE" (normal) or "BAAD" (chkdisk error for entry)
  - flags for the type of file (file or directory) and if the entry is "in use"
  - link count (number of directories have entries MFT entry)
  - offset to the first attribute
  - information for the fixup array

Master File Table (MFT)

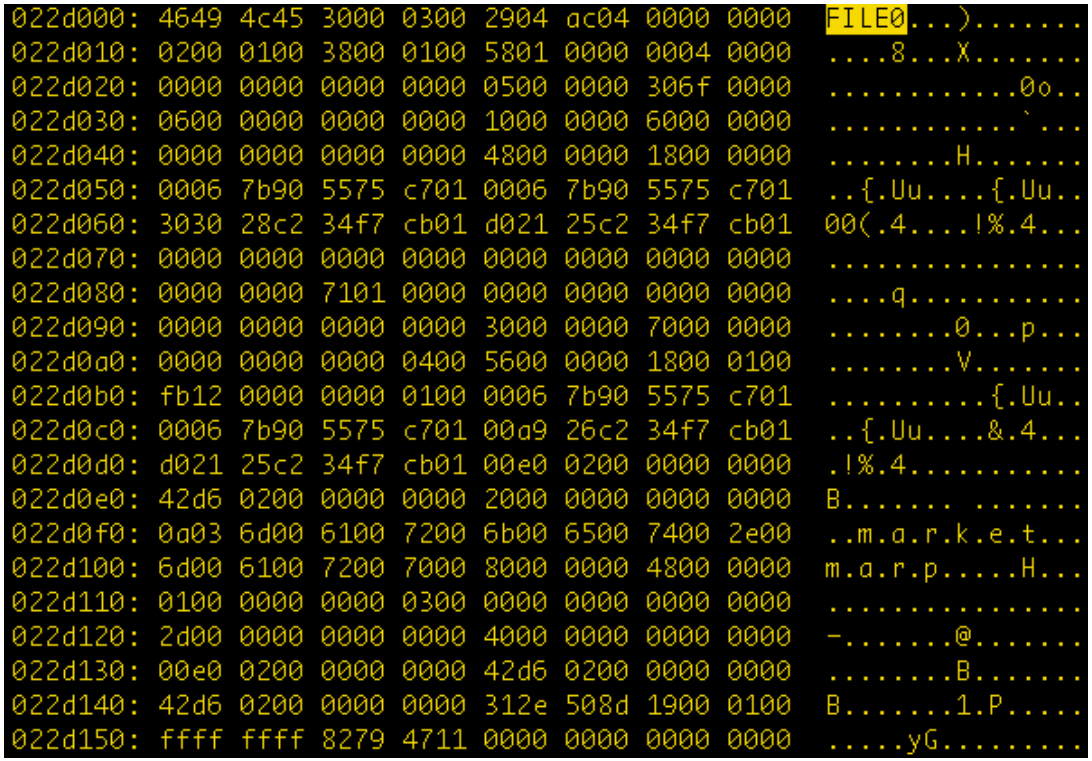
- Attributes are either resident or non-resident
- This is determined from the attribute header
- Resident items are contained entirely in the MFT entry
- Non-resident items are contained in more than one MFT entry

Master File Table (MFT) [4]



Methodology

- Scan for signatures “FILE” and “BAAD”
- Parse out the MFT entry



Master File Table (MFT)

```
'MFT_FILE_RECORD': [ 0x400, {
    'Signature': [ 0x0, ['unsigned int']],
    'fixup_array_offset': [ 0x4, ['unsigned short']],
    'num_fixup_entries': [ 0x6, ['unsigned short']],
    'LSN': [ 0x8, ['unsigned long long']],
    'sequence_value': [ 0x10, ['unsigned short']],
    'link_count': [ 0x12, ['unsigned short']],
    'first_attribute_offset': [0x14, ['unsigned short']],
    'flags': [0x16, ['unsigned short']],
    'entry_used_size': [0x18, ['unsigned int']],
    'entry_allocated_size': [0x1c, ['unsigned int']],
    'file_ref_base_record': [0x20, ['unsigned long long']],
    'next_attribute_id': [0x28, ['unsigned short']],
    'record_num': [0x2c, ['unsigned long']],
    'fix_up_array': lambda x: obj.Object("Array", offset = x.obj_offset +
x.fixup_array_offset, count = x.num_fixup_entries, vm = x.obj_vm,
                                    target = obj.Curry(obj.Object,"unsigned
short")),
    'resident_attributes': lambda x : obj.Object("RESIDENT_ATTRIBUTE", offset =
x.obj_offset + x.first_attribute_offset, vm=x.obj_vm),
    'non_resident_attributes': lambda x : obj.Object("NON_RESIDENT_ATTRIBUTE",
offset = x.obj_offset + x.first_attribute_offset, vm=x.obj_vm),
 }],
```

Master File Table (MFT)

```
'ATTRIBUTE_HEADER': [ 0x10, {
    'type': [0x0, ['int']],
    'length': [0x4, ['int']],
    'non_resident_flag': [0x8, ['unsigned
char']],
    'name_length': [0x9, ['unsigned char']],
    'name_offset': [0xa, ['unsigned
short']],
    'flags': [0xc, ['unsigned short']],
    'attribute_id': [0xe, ['unsigned
short']],
}],
```

Master File Table (MFT)

```
'RESIDENT_ATTRIBUTE': [0x16, {
    'header': [0x0, ['ATTRIBUTE_HEADER']],
    'content_size': [0x10, ['unsigned int']], #relative to the beginning of the attribute
    'content_offset': [0x14, ['unsigned short']],
    'std_info': lambda x : obj.Object("STANDARD_INFORMATION", offset=x.obj_offset +
x.content_offset, vm=x.obj_vm),
    'FILE_NAME': lambda x : obj.Object("FILE_NAME", offset=x.obj_offset + x.content_offset,
vm=x.obj_vm),
    'obj_id': lambda x : obj.Object("OBJECT_ID", offset=x.obj_offset + x.content_offset,
vm=x.obj_vm),
    'attr_list':lambda x : obj.Object("ATTRIBUTE_LIST", offset=x.obj_offset + x.content_offset,
vm=x.obj_vm),
}],

'NON_RESIDENT_ATTRIBUTE': [0x40, {
    'header': [0x0, ['ATTRIBUTE_HEADER']],
    'starting_VCN': [0x10, ['unsigned long long']],
    'ending_VCN': [0x18, ['unsigned long long']],
    'runlist_offset': [0x20, ['unsigned short']],
    'compression_unit_size': [0x22, ['unsigned short']],
    'unused': [0x24, ['int']],
    'allocated_attribute_size': [0x28, ['unsigned long long']],
    'actual_attribute_size': [0x30, ['unsigned long long']],
    'initialized_attribute_size': [0x38, ['unsigned long long']],
}],


Etc….
```

MFT – Metadata

- We have several timestamps associated with various attributes. Below is a description[4]

- Creation Time - The time the file was created
- Modified Time - The time the content of the $DATA or $INDEX attributes were last modified
- MFT Modified Time - The time the metadata of the file was last modified (not shown in windows under properties)
- Accessed Time - The time that the content of the file was last accessed

MFT - $STANDARD_INFO

- This attribute exists for all files and directories and contains core metadata [4]. Things you can find:
  - Four timestamps (shown under "Metadata" above) for various file events
  - File ownership
  - Security info (Security ID is not a SID, but an offset into $SECURITY)
  - Quota info
  - Flags that indicate the type of file and how if it is encrypted, compressed, hidden etc
  - Lots of MS applications rely on this attribute even though it is not an essential attribute [4].

MFT - $STANDARD_INFO

```
'STANDARD_INFORMATION': [0x48, {
    'CreationTime': [0x0, ['WinTimeStamp', {}]],
    'AlteredTime': [0x8, ['WinTimeStamp', {}]],
    'MFT_AlteredTime': [0x10, ['WinTimeStamp', {}]],
    'FileAccessedTime': [0x18, ['WinTimeStamp', {}]],
    'flags': [0x20, ['int']],
    'max_version_number': [0x24, ['unsigned int']],
    'version_number': [0x28, ['unsigned int']],
    'class_ID': [0x2c, ['unsigned int']],
    'owner_ID': [0x30, ['unsigned int']],
    'security_ID': [0x34, ['unsigned int']],
    'quota_charged': [0x38, ['unsigned long long']],
    'USN': [0x40, ['unsigned long long']],
    'next_attribute': [0x48, ['RESIDENT_ATTRIBUTE']],
}],
```

MFT - $FILE_NAME

- The $FILE_NAME is used to store the parent directory information and the filename [4]. There is no essential information when it is used in an MFT entry, but there is when it is used in a directory index [4].
- The parent directory is for file reference: upper two bytes are the sequence number and lower six bytes are the MFT entry.
- Timestamps are the same as those for $STANDARD_INFO
- Namespace denotes what type of characters make up the filename
- Flags are the same as described for $STANDARD_INFO

MFT - $FILE_NAME

```
'FILE_INFO': [None, {
    'parent_directory': [0x0, ['unsigned long long']],
    'FileCreationTime': [0x8, ['WinTimeStamp', {}]],
    'FileModificationTime': [0x10, ['WinTimeStamp', {}]],
    'MFT_AlteredTime': [0x18, ['WinTimeStamp', {}]],
    'FileAccessedTime': [0x20, ['WinTimeStamp', {}]],
    'allocated_file_size': [0x28, ['unsigned long long']],
    'real_file_size': [0x30, ['unsigned long long']],
    'flags': [0x38, ['unsigned int']],
    'reparse_value': [0x3c, ['unsigned int']],
    'name_length': [0x40, ['unsigned char']],
    'namespace': [0x41, ['unsigned char']],
    'name': [0x42, ['array', lambda x : x.name_length *2 , ['unsigned char']]],
}],
```

MFT - $FILE_NAME

```
def get_full_path(self, fileinfo):
    parent = "-1"
    parent_id = fileinfo.parent_directory & 0xffffff
    path = self.remove_unprintable(fileinfo.get_name())
    while parent != {}:
        parent = MFT_PATHS_FULL.get(int(parent_id), {})
        if parent == {} or parent["filename"] == "":
            return path
        path = parent["filename"] + "\\" + path
        parent_id = parent["parent_directory"] & 0xffffff
    return path
```

MFT - $DATA

- $DATA attributes contain no structure (but do contain an attribute header of course)
- If resident they contain the file content, however if the file content is over 700 bytes content is non-resident most likely [4].
- It is often the last attribute of an MFT entry, but it is possible that this may not be the case. Directories can also have $DATA attributes in addition to index attributes [4].

Example: cve2011_0611.dmp

- Looking at the public sample from
- http://sempersecurus.blogspot.com/2011/04/using-volatility-to-study-cve-2011-6011.html

- Flash exploit embedded in a word document

Example: cve2011_0611.dmp

- We can recover the path for the exploit file using the `filescan` plugin:

0x02208ad8 1 1 R--rw- \Device \HarddiskVolume1\Documents and Settings\1\My Documents\FLASH WITH DOC Disentangling Industrial Policy and Competition Policy

- But we don’t know when the file hit the system

Example: cve2011_0611.dmp

```
MFT entry found at offset 0x9790800
Type: In Use & Directory
Record Number: 2230
Number of fixup array vals 3
Link count: 2
Sequence Value: 0x6
Fixup Array: 0x4 0x4 0x0

$STANDARD_INFO
Creation             Modified             MFT Altered          Access Date          Type
-------------------- -------------------- -------------------- -------------------- ----
2011-04-10 22:27:54  2011-04-10 22:29:50  2011-04-10 22:29:50  2011-04-10 22:29:51  Unknown Type

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2011-04-10 22:27:54  2011-04-10 22:27:54  2011-04-10 22:27:54  2011-04-10 22:27:54  FLASHW~1
Full Path: DOCUME~1\1\MYDOCU~1\FLASHW~1

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2011-04-10 22:27:54  2011-04-10 22:27:54  2011-04-10 22:27:54  2011-04-10 22:27:54  FLASH WITH DOC Disentangling Industrial Policy
and Competition Policy
Full Path: DOCUME~1\1\MYDOCU~1\FLASH WITH DOC Disentangling Industrial Policy and Competition Policy

$OBJECT_ID
Object ID: c173d4d7-c163-e011-923d-0800275d6d42
Birth Volume ID: 90000000-5800-0000-0004-180000000700
Birth Object ID: 38000000-2000-0000-2400-490033003000
Birth Domain ID: 30000000-0100-0000-0010-000001000000
*************************************************************************
```

Example: cve2011_0611.dmp

```
MFT entry found at offset 0x33af000
Type: In Use & File
Record Number: 2304
Number of fixup array vals 3
Link count: 2
Sequence Value: 0x6
Fixup Array: 0x2 0x0 0x0

$STANDARD_INFO
Creation             Modified             MFT Altered          Access Date          Type
-------------------- -------------------- -------------------- -------------------- ----
2011-04-10 22:28:29  2011-04-10 22:30:20  2011-04-10 22:30:20  2011-04-10 22:30:20  Archive

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2011-04-10 22:28:29  2011-04-10 22:30:20  2011-04-10 22:30:20  2011-04-10 22:30:20  FLASHW~1.LNK
Full Path: DOCUME~1\1\APPLIC~1\MICROS~1\Office\Recent\FLASHW~1.LNK

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2011-04-10 22:28:29  2011-04-10 22:30:20  2011-04-10 22:30:20  2011-04-10 22:30:20  FLASH WITH DOC
Disentangling Industrial Policy and Competition Policy.LNK
Full Path: DOCUME~1\1\APPLIC~1\MICROS~1\Office\Recent\FLASH WITH DOC Disentangling Industrial Policy and
Competition Policy.LNK

$DATA
non-resident

*******************************************************************************
```

Example: GrrCon challenge

```
***************************************************************************
MFT entry found at offset 0x14c42000
Type: In Use & File
Record Number: 12024
Number of fixup array vals 3
Link count: 2
Sequence Value: 0x4
Fixup Array: 0x3 0x0 0x0

$STANDARD_INFO
Creation             Modified             MFT Altered          Access Date          Type
-------------------- -------------------- -------------------- -------------------- ----
2012-04-28 01:59:22  2012-04-28 01:59:22  2012-04-28 01:59:22  2012-04-28 01:59:22  Archive & Content not indexed

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2012-04-28 01:59:22  2012-04-28 01:59:22  2012-04-28 01:59:22  2012-04-28 01:59:22  SWING-~1.PF
Full Path: WINDOWS\Prefetch\SWING-~1.PF

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2012-04-28 01:59:22  2012-04-28 01:59:22  2012-04-28 01:59:22  2012-04-28 01:59:22  SWING-MECHANICS.DOC
[1].EXE-013CEA10.pf
Full Path: WINDOWS\Prefetch\SWING-MECHANICS.DOC[1].EXE-013CEA10.pf
```

Example: GrrCon challenge

```
**************************************************************************
MFT entry found at offset 0xdc8b800
Type: In Use & File
Record Number: 11978
Number of fixup array vals 3
Link count: 1
Sequence Value: 0x4
Fixup Array: 0x3 0x0 0x0

$STANDARD_INFO
Creation             Modified             MFT Altered          Access Date          Type
-------------------- -------------------- -------------------- -------------------- ----
2012-04-28 02:02:38  2012-04-28 02:02:39  2012-04-28 02:02:39  2012-04-28 02:02:39  Archive

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2012-04-28 02:02:38  2012-04-28 02:02:38  2012-04-28 02:02:38  2012-04-28 02:02:38  w.exe
Full Path: WINDOWS\system32\systems\w.exe

$DATA
non-resident

**************************************************************************
```

Example: GrrCon challenge

```
**************************************************************************
MFT entry found at offset 0x15938c00
Type: In Use & File
Record Number: 12031
Number of fixup array vals 3
Link count: 1
Sequence Value: 0x3
Fixup Array: 0x9 0x0 0x0

$STANDARD_INFO
Creation             Modified             MFT Altered          Access Date          Type
-------------------- -------------------- -------------------- -------------------- ----
2012-04-28 02:01:54  2012-04-28 02:01:54  2012-04-28 02:01:54  2012-04-28 02:10:14  Archive

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2012-04-28 02:01:54  2012-04-28 02:01:54  2012-04-28 02:01:54  2012-04-28 02:01:54  g.exe
Full Path: WINDOWS\system32\systems\g.exe

$DATA
non-resident

**************************************************************************
```

Example: GrrCon challenge

```
***************************************************************************
MFT entry found at offset 0x1eb0a8b0
Type: In Use & File
Record Number: 12045
Number of fixup array vals 3
Link count: 2
Sequence Value: 0x3
Fixup Array: 0x1 0x0 0x0

$STANDARD_INFO
Creation             Modified             MFT Altered          Access Date          Type
-------------------- -------------------- -------------------- -------------------- ----
2012-04-28 02:07:44  2012-04-28 02:07:44  2012-04-28 02:07:44  2012-04-28 02:07:44  Archive

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2012-04-28 02:07:44  2012-04-28 02:07:44  2012-04-28 02:07:44  2012-04-28 02:07:44  CONFID~4.PDF
Full Path: WINDOWS\system32\systems\1\CONFID~4.PDF

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2012-04-28 02:07:44  2012-04-28 02:07:44  2012-04-28 02:07:44  2012-04-28 02:07:44  confidential4.pdf
Full Path: WINDOWS\system32\systems\1\confidential4.pdf

$DATA
non-resident

***************************************************************************
```

Example: GrrCon challenge

```
**************************************************************************
MFT entry found at offset 0x15938800
Type: In Use & File
Record Number: 12030
Number of fixup array vals 3
Link count: 1
Sequence Value: 0x3
Fixup Array: 0x9 0x0 0x0

$STANDARD_INFO
Creation             Modified             MFT Altered          Access Date          Type
-------------------- -------------------- -------------------- -------------------- ----
2012-04-28 02:01:43  2012-04-28 02:01:43  2012-04-28 02:01:43  2012-04-28 02:01:43  Archive

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2012-04-28 02:01:43  2012-04-28 02:01:43  2012-04-28 02:01:43  2012-04-28 02:01:43  f.txt
Full Path: WINDOWS\system32\systems\f.txt

$DATA
0x00000000: 6f 70 65 6e 20 36 36 2e 33 32 2e 31 31 39 2e 33   open.66.32.119.3
0x00000010: 38 0d 0a 6a 61 63 6b 0d 0a 32 61 77 65 73 30 6d   8..jack..2awes0m
0x00000020: 65 0d 0a 6c 63 64 20 63 3a 5c 57 49 4e 44 4f 57   e..lcd.c:\WINDOW
0x00000030: 53 5c 53 79 73 74 65 6d 33 32 5c 73 79 73 74 65   S\System32\syste
0x00000040: 6d 73 0d 0a 63 64 20 20 2f 68 6f 6d 65 2f 6a 61   ms..cd../home/ja
0x00000050: 63 6b 0d 0a 62 69 6e 61 72 79 0d 0a 6d 70 75 74   ck..binary..mput
0x00000060: 20 22 2a 2e 74 78 74 22 0d 0a 64 69 73 63 6f 6e   ."*.txt"..discon
0x00000070: 6e 65 63 74 0d 0a 62 79 65 0d 0a                  nect..bye..
**************************************************************************
```

Check out the Data!

Example: GrrCon challenge

```
$ cat f.raw|xxd -r > f.txt
$ cat f.txt
open 66.32.119.38
jack
2awes0me
lcd c:\WINDOWS\System32\systems
cd  /home/jack
binary
mput "*.txt"
disconnect
bye
```

Example: GrrCon challenge

```
$ grep systems grr_mft
Full Path: WINDOWS\system32\systems\1\CONFID~3.PDF
Full Path: WINDOWS\system32\systems\1\confidential3.pdf
Full Path: WINDOWS\system32\systems\1\CONFID~4.PDF
Full Path: WINDOWS\system32\systems\1\confidential4.pdf
Full Path: WINDOWS\system32\systems\w.exe
Full Path: WINDOWS\system32\systems\1\CO20EF~1.PDF
Full Path: WINDOWS\system32\systems\1\confidential5.pdf
2012-04-28 02:01:03  2012-04-28 02:01:03  2012-04-28 02:01:03  2012-04-28 02:01:03  systems
Full Path: WINDOWS\system32\systems
Full Path: WINDOWS\system32\systems\f.txt
Full Path: WINDOWS\system32\systems\g.exe
Full Path: WINDOWS\system32\systems\p.exe
Full Path: WINDOWS\system32\systems\r.exe
Full Path: WINDOWS\system32\systems\sysmon.exe
Full Path: WINDOWS\system32\systems\p.exe
Full Path: WINDOWS\system32\systems\r.exe
Full Path: WINDOWS\system32\systems\sysmon.exe
Full Path: WINDOWS\system32\systems\1
Full Path: WINDOWS\system32\systems\1\CONFID~4.PDF
Full Path: WINDOWS\system32\systems\1\confidential4.pdf
Full Path: WINDOWS\system32\systems\w.exe
```

Timestomping – SetMACE

- Experiment: use SetMACE to change the MAC times on a file and see if it changed in memory

- http://code.google.com/p/mft2csv/downloads/detail?name=SetMACE_v1006.zip&can=2&q=

Timestomping – SetMACE

- Setting the $FILE_NAME timestamps didn't affect the MFT entry in memory in the first experiment
- Let it run 5 minutes and re-accessed the file.

$STANDARD_INFO

Creation	Modified	MFT Altered	Access Date	Type
2012-06-18 19:52:30	2012-06-18 19:52:31	2012-06-18 19:52:44	2012-06-18 19:52:44	Archive

$FILE_NAME

Creation	Modified	MFT Altered	Access Date	Name/Path
2012-06-18 19:52:30	2012-06-18 19:52:30	2012-06-18 19:52:30	2012-06-18 19:52:30	POISON~1.PY

Full Path: DOCUME~1\user\Desktop\POISON~1.PY

$FILE_NAME

Creation	Modified	MFT Altered	Access Date	Name/Path
2012-06-18 19:52:30	2012-06-18 19:52:30	2012-06-18 19:52:30	2012-06-18 19:52:30	poison_ivy.py

Full Path: DOCUME~1\user\Desktop\poison_ivy.py

Timestomping – SetMACE

- Changing the $FILE_NAME and $STANDARD_INFO timestamps reflected in memory with only a slight delay
- Also accessing the file again and modifying it reflected in memory with a slight delay
- Also the offsets in memory remained the same

Timestomping – SetMACE

```
$STANDARD_INFO
Creation             Modified             MFT Altered          Access Date          Type
-------------------- -------------------- -------------------- -------------------- ----
2000-01-01 00:00:00  2000-01-01 00:00:00  2000-01-01 00:00:00  2000-01-01 00:00:00   Archive

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2000-01-01 00:00:00  2000-01-01 00:00:00  2000-01-01 00:00:00  2000-01-01 00:00:00  POISON~1.PY
Full Path: DOCUME~1\user\Desktop\POISON~1.PY

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2000-01-01 00:00:00  2000-01-01 00:00:00  2000-01-01 00:00:00  2000-01-01 00:00:00
poison_ivy.py
Full Path: DOCUME~1\user\Desktop\poison_ivy.py
```

Timestomping – SetMACE

```
***************************************************************************
MFT entry found at offset 0x1bf3a800
Type: In Use & File
Record Number: 12226
Number of fixup array vals 3
Link count: 2
Sequence Value: 0x3
Fixup Array: 0x14 0x0 0x0

$STANDARD_INFO
Creation            Modified            MFT Altered         Access Date         Type
------------------- ------------------- ------------------- ------------------- ----
2000-01-01 00:00:00  2000-01-01 00:00:00  2000-01-01 00:00:00  2012-09-25 17:55:45  Archive

$FILE_NAME
Creation            Modified            MFT Altered         Access Date         Name/Path
------------------- ------------------- ------------------- ------------------- ---------
2000-01-01 00:00:00  2000-01-01 00:00:00  2000-01-01 00:00:00  2000-01-01 00:00:00  POISON~1.PY
Full Path: DOCUME~1\user\Desktop\POISON~1.PY

$FILE_NAME
Creation            Modified            MFT Altered         Access Date         Name/Path
------------------- ------------------- ------------------- ------------------- ---------
2000-01-01 00:00:00  2000-01-01 00:00:00  2000-01-01 00:00:00  2000-01-01 00:00:00  poison_ivy.py
Full Path: DOCUME~1\user\Desktop\poison_ivy.py
```

Timestomping – SetMACE

- Deleting the file also showed up in memory in less than a minute's time

```
***************************************************************************
MFT entry found at offset 0x2037800
Type: File
Record Number: 12226
Number of fixup array vals 3
Link count: 1
Sequence Value: 0x4
Fixup Array: 0x1c 0x1147 0x0

$STANDARD_INFO
Creation            Modified            MFT Altered         Access Date         Type
------------------- ------------------- ------------------- ------------------- ----
2000-01-01 00:00:00  2012-09-25 18:08:25  2012-09-25 18:08:40  2012-09-25 18:08:25  Archive

$FILE_NAME
Creation            Modified            MFT Altered         Access Date         Name/Path
------------------- ------------------- ------------------- ------------------- --------
2000-01-01 00:00:00  2012-09-25 18:08:25  2012-09-25 18:08:25  2012-09-25 18:08:25  Dc2.py
Full Path: RECYCLER\S-1-5-~1\Dc2.py
```

Caveat: Multiple Volumes

- On a system running with multiple NTFS volumes MFT records may conflict with one another in memory resulting in funky paths
- This is a result of duplicate record numbers
- No separate $MFT's found on separate volumes

- It's all in memory

Caveat: Multiple Volumes

```
***************************************************************************
MFT entry found at offset 0x72ec00
Type: In Use & File
Record Number: 139
Number of fixup array vals 3
Link count: 1
Sequence Value: 0x1
Fixup Array: 0x1f 0x0 0x0

$STANDARD_INFO
Creation             Modified             MFT A                          Type
-------------------- -------------------- -------- -------------------- ----
2001-08-23 12:00:00  2001-08-23 12:00:00  2010-06-0 09:15:08  2010-11-11 14:46:55  Compressed &
Archive

$FILE_NAME
Creation             Modified             MFT Altered          Access Date          Name/Path
-------------------- -------------------- -------------------- -------------------- ---------
2010-06-04 09:10:07  2010-06-04 09:10:07  2010-06-04 09:10:07  2010-06-04 09:10:07  kdcom.dll
Full Path: NEWTEX~1.TXT\kdcom.dll
***************************************************************************
```

TrueCrypt volume vs System

Questions?

Coordinates:

Volatility:

http://volatility-labs.blogspot.com/

http://code.google.com/p/volatility/

Email:

jamie.levy@gmail.com

Twitter:

@gleeda

References

- [1] http://thestarman.narod.ru/asm/mbr/Win2kmbr.htm
- [2] http://thestarman.narod.ru/asm/mbr/VistaMBR.htm
- [3] http://thestarman.narod.ru/asm/mbr/W7MBR.htm
- [4] File System Forensic Analysis, Brian Carrier