

Physical Memory File Extraction Based on File Object Analysis

Youngbok Kang(K-dupe)

with Ph.d Hyunuk Hwang and Ph.d Kibom Kim

Chonnam National University | SSRC

Content

- **Background**
- **File Object Analysis**
- **File Cache Analysis**
- **DEMO**
- **Q/A**

Outline

▪ Who am I ?

- Ph.d Student at Chonnam National University

▪ Objectives

- Introducing a Physical Memory Forensic
 - Memory dump
 - Advantages
- Window Physical Memory Analysis
 - Analysis of window file object
 - Analysis of memory mapped file data
- Showing some demos

What can be found in Memory

- **Running process information**
 - thread, module, handle...
- **Registry hives**
- **Network Information**
 - Packet, Socket info...
- **Malware and Rootkit detection**
- **Password key and other information**
- **File Data**
 - Running process File data, Cached File Data

Memory Forensic Advantages

- **Analyze and track active on the system**
 - Running process analysis
 - Access file, Network Info etc...

- **Collect evidence that cannot be found in hard disk**
 - User Information
 - Messenger msg, chat data, internet activities
 - Password key data
 - Master Key
 - Original file extraction
 - When encrypted volume was mounted

Background

■ Physical Memory Dump

- Live Dump
 - Easy and general method
 - Tool : Win32/64DD, Winpmem, Dumpit etc...

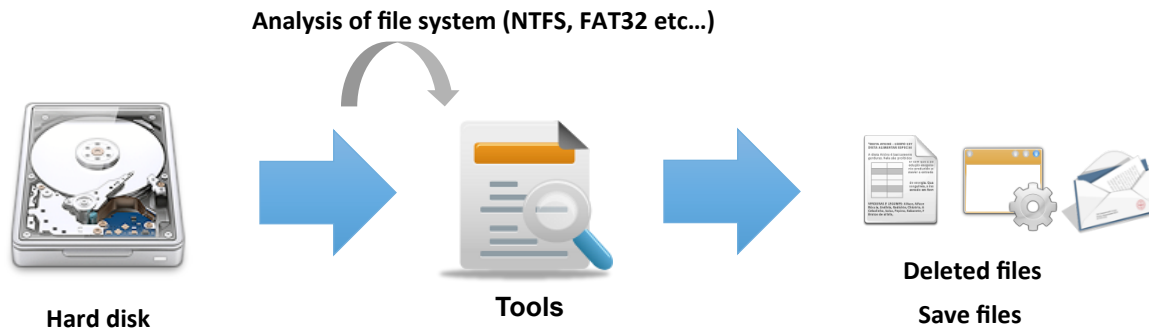
- Crash Dump
 - Window Kernel Crash Dump
 - %SystemRoot%Memory.dmp

- Hibernate Dump
 - It contains a full dump of the memory
 - %SystemRoot%hiberfil.sys

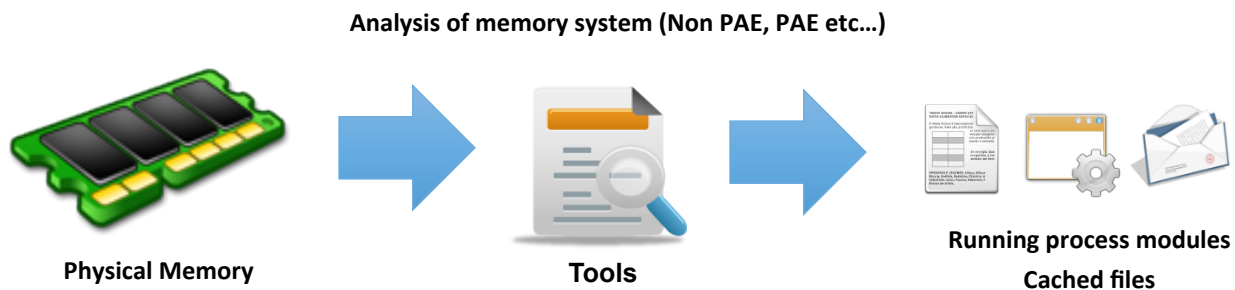
Background

■ File Extraction

- Hard-Disk



- Physical Memory Dump



Research trend

▪ Analysis of Memory mapped file data

- FileObject Analysis

- Blackhat USA 2011-Physical Memory Forensics for Files and Cache
- Forensic memory analysis : Files mapped in memory - 2008

- VAD Analysis

- The VAD tree : A process-eye view of physical memory - 2007

Research trend

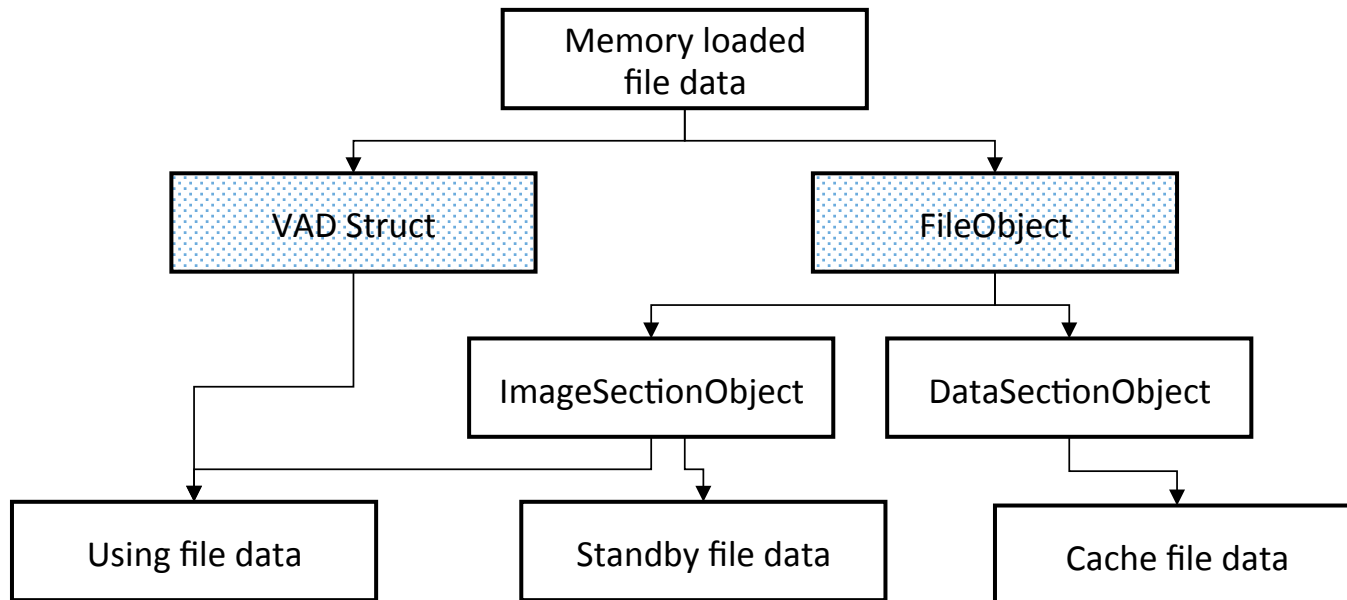
▪ Extracting file data from memory dump

- Executable file (running process)
 - VAD Struct analysis
 - Tool : Volatility, Rekall, Responder, Memoryze (FileObject, VAD)
- Normal file
 - File Carving (file size < 4KB)
- Cache file
 - File I/O analysis, VACB(Virtual Address Control Block) struct analysis

Memory mapped file

■ File data storage areas

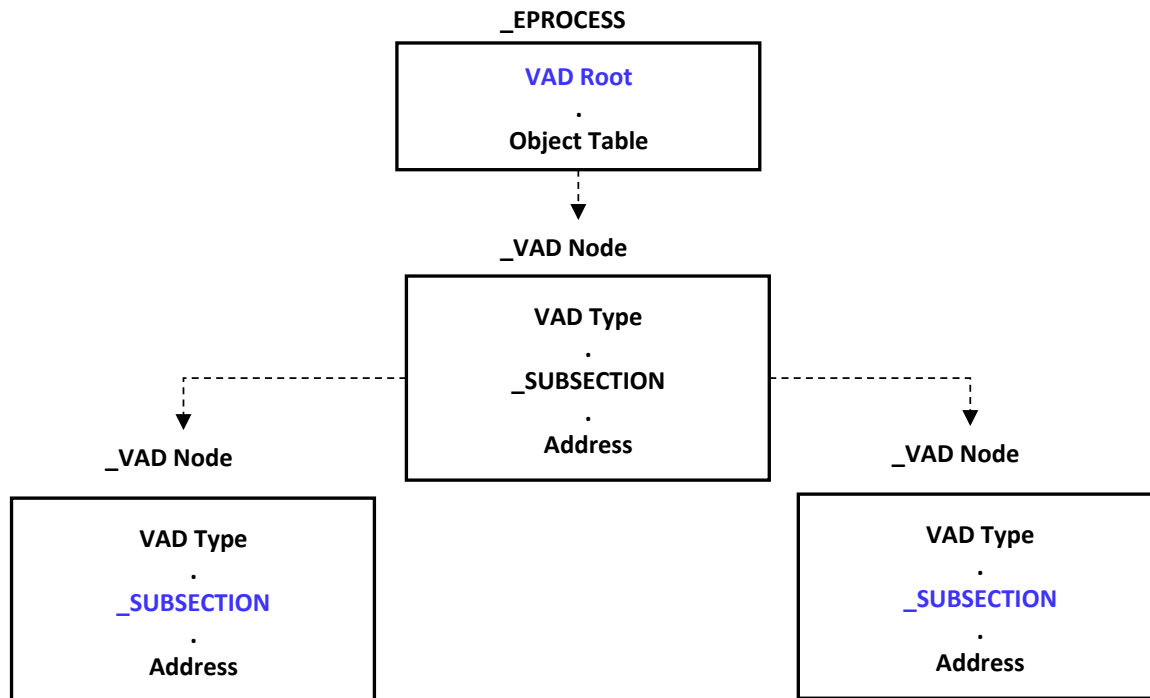
- VAD struct
- FileObject struct



Memory mapped file

■ VAD (Virtual Address Descriptor) (1/2)

- Describe memory ranges used by a process
- `_MMVAD` struct
 - Consists of a binary tree



```
kd> dt _MMVAD
nt!_MMVAD
+0x000 u1           : <unnamed-tag>
+0x008 LeftChild   : Ptr64 _MMVAD
+0x010 RightChild  : Ptr64 _MMVAD
+0x018 StartingVpn : Uint8B
+0x020 EndingVpn   : Uint8B
+0x028 u           : <unnamed-tag>
+0x030 PushLock    : _EX_PUSH_LOCK
+0x038 u5          : <unnamed-tag>
+0x040 u2          : <unnamed-tag>
+0x048 Subsection  : Ptr64 _SUBSECTION
+0x048 MappedSubsection : Ptr64 _MSUBSECTION
+0x050 FirstPrototypePte : Ptr64 _MMPTE
+0x058 LastContiguousPte : Ptr64 _MMPTE
+0x060 ViewLinks   : LIST_ENTRY
+0x070 VadsProcess  : Ptr64 _EPROCESS
```

`_MMVAD` struct

Memory mapped file

■ VAD (Virtual Address Descriptor) (2/2)

- VAD -> File data loaded
 - `_SUBSECTION` is Not NULL
- VAD -> Heap/Stack
 - `_SUBSECTION` is NULL

```
nt!_MMVAD
+0x000 ul          : <unnamed-tag>
+0x008 LeftChild   : 0xfffffa80`0dc55330 _MMVAD
+0x010 RightChild  : (null)
+0x018 StartingVpn : 0x8b60
+0x020 EndingVpn   : 0x8b60
+0x028 u           : <unnamed-tag>
+0x030 PushLock    : _EX_PUSH_LOCK
+0x038 u5          : <unnamed-tag>
+0x040 u2          : <unnamed-tag>
+0x048 Subsection  : (null)
+0x048 MappedSubsection : (null)
+0x050 FirstPrototypePte : (null)
+0x058 LastContiguousPte : (null)
+0x060 ViewLinks   : _LIST_ENTRY [ 0x00000000`00000000 - 0x0 ]
+0x070 VadsProcess : (null)
```

`_MMVAD` Heap/Stack

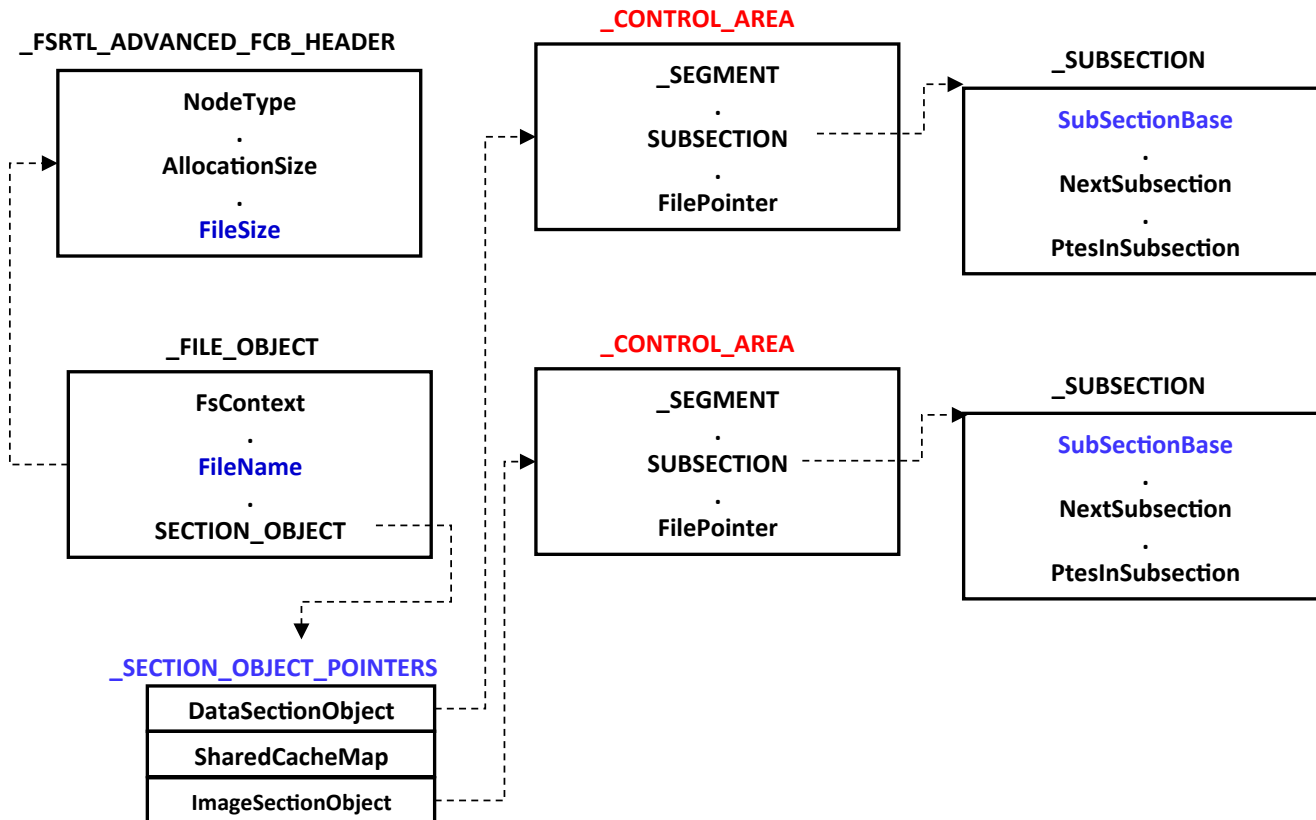
```
kd> dt _MMVAD fffffa800cf54f70
nt!_MMVAD
+0x000 ul          : <unnamed-tag>
+0x008 LeftChild   : 0xfffffa80`0ef487b0 _MMVAD
+0x010 RightChild  : 0xfffffa80`0ef3e170 _MMVAD
+0x018 StartingVpn : 0x8ba0
+0x020 EndingVpn   : 0x8c1f
+0x028 u           : <unnamed-tag>
+0x030 PushLock    : _EX_PUSH_LOCK
+0x038 u5          : <unnamed-tag>
+0x040 u2          : <unnamed-tag>
+0x048 Subsection  : 0xfffffa80`0efba620 _SUBSECTION
+0x048 MappedSubsection : 0xfffffa80`0efba620 _MSUBSECTION
+0x050 FirstPrototypePte : 0xfffff8a0`01fa4a10 _MMPTE
+0x058 LastContiguousPte : 0xfffff8a0`01fa4a20 _MMPTE
+0x060 ViewLinks   : _LIST_ENTRY [ 0xfffffa80`0efba610 - 0xfffffa80`0efba610 ]
+0x070 VadsProcess : 0xfffffa80`0eb1b061 _EPROCESS
```

`_MMVAD` File data loaded

Memory mapped file

FileObject struct

- Managing file information



Memory mapped file

▪ FileObject Analysis

- `_SECTION_OBJECT_POINTERS`

- DataSectionObject

- Present on binary memory mapped files and normal files
 - File : .exe .dll .jpg .pdf ...

- SharedCacheMap

- ReadWrite I/O Data
 - VACB(Virtual Address Control Block) Struct

- ImageSectionObject

- Present on binary memory mapped files
 - File : .exe .dll

Memory mapped file

▪ FileObject Analysis

- `_SECTION_OBJECT_POINTERS`

- Executable file

- ImageSectionObject, DataSectionObject

```
kd> dt _SECTION_OBJECT_POINTERS 0xfffffa80`0cf15a08
ntdll!_SECTION_OBJECT_POINTERS
+0x000 DataSectionObject : 0xfffffa80`0cf64840 Void
+0x008 SharedCacheMap    : (null)
+0x010 ImageSectionObject : 0xfffffa80`0cf15440 Void
```

- Normal file

- DataSectionObject

```
kd> dt _SECTION_OBJECT_POINTERS 0xfffffa80`0ead0ea8
ntdll!_SECTION_OBJECT_POINTERS
+0x000 DataSectionObject : 0xfffffa80`0edc11d0 Void
+0x008 SharedCacheMap    : (null)
+0x010 ImageSectionObject : (null)
```

Memory mapped file

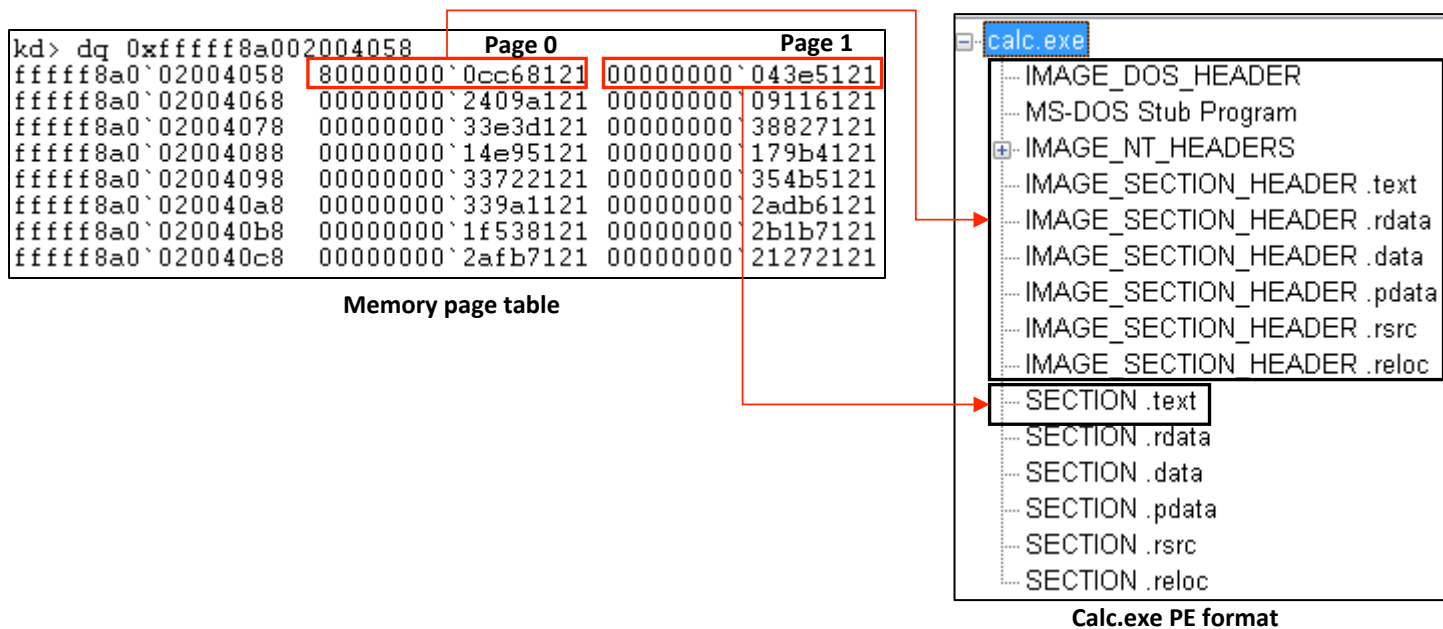
File data management

- ImageSectionObject

- ImageSectionObject -> _CONTROL_AREA -> _SUBSECTION

- _SUBSECTION -> SubsectionBase

- Physical address of file data saved



Memory mapped file

- File data management

- DataSectionObject

- DataSectionObject -> _CONTROL_AREA -> _SUBSECTION
- _SUBSECTION -> SubsectionBase
 - Physical address of file data saved

Offset : 0 calc.exe File data

```

00000000 4D 5A 90 00 03 00 00 04 00 00 FF FF 90 00 MZ
00000001 85 90 00 00 00 00 00 40 00 00 00 00 00
00000002 00 00 00 00 00 00 00 00 00 00 00 00 00
00000003 00 00 00 00 00 00 00 00 00 00 F0 00 00
00000004 05 1F BA 0E 00 BA 09 00 21 88 01 4C 02 21 54 68
00000005 69 73 20 70 72 6F 67 72 61 60 20 63 81 8E 8E 8F is program cannot
00000006 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 52 20 i be run in DOS
00000007 69 6F 64 25 03 02 0A 24 00 00 00 00 00 mode $
00000008 E3 CA 74 E8 A4 AB 1A B5 A4 AB 1A B5 A4 AB 1A B5 t
00000009 AD D3 9F 85 A6 AB 1A B5 AD D3 9E 85 9A AB 1A B5
0000000A A4 AB 1B 85 30 AA 1A B5 AD D3 99 85 95 AB 1A B5 0
0000000B AD D3 99 85 83 AB 1A B5 AD D3 90 85 F3 AB 1A B5
0000000C AD D3 9E 85 A6 AB 1A B5 AD D3 98 85 A6 AB 1A B5
0000000D 52 89 63 88 A4 AB 1A B5 00 00 00 00 00 00 Rich
0000000E 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000F 50 45 00 00 58 86 06 04 C3 58 4A 00 00 00 FE
00000010 00 00 00 F0 00 00 00 06 02 08 00 00 0E 96 00
00000011 00 F2 07 00 00 00 00 88 89 01 00 10 10 00
00000012 00 00 00 01 01 00 00 00 00 00 00 00 00
00000013 06 00 01 00 06 00 01 06 00 01 00 00 00 00
00000014 00 30 0E 00 00 00 00 00 00 00 00 00 00
00000015 00 00 00 00 00 00 00 00 00 00 00 00 00
    
```

Page 0

```

kd> dq fffff8a00239fb60
fffff8a0`03ddc000 00000000`384478c0 fa800ceb`14c004c0
fffff8a0`03ddc010 fa800ceb`14c004c0 fa800ceb`14c004c0
fffff8a0`03ddc020 fa800ceb`14c004c0 fa800ceb`14c004c0
fffff8a0`03ddc030 fa800ceb`14c004c0 fa800ceb`14c004c0
fffff8a0`03ddc040 fa800ceb`14c004c0 fa800ceb`14c004c0
fffff8a0`03ddc050 fa800ceb`14c004c0 fa800ceb`14c004c0
fffff8a0`03ddc060 fa800ceb`14c004c0 fa800ceb`14c004c0
fffff8a0`03ddc070 fa800ceb`14c004c0 fa800ceb`14c004c0
    
```

Memory page table

Page 10

Offset : 0x1000

Offset : 0xA000

```

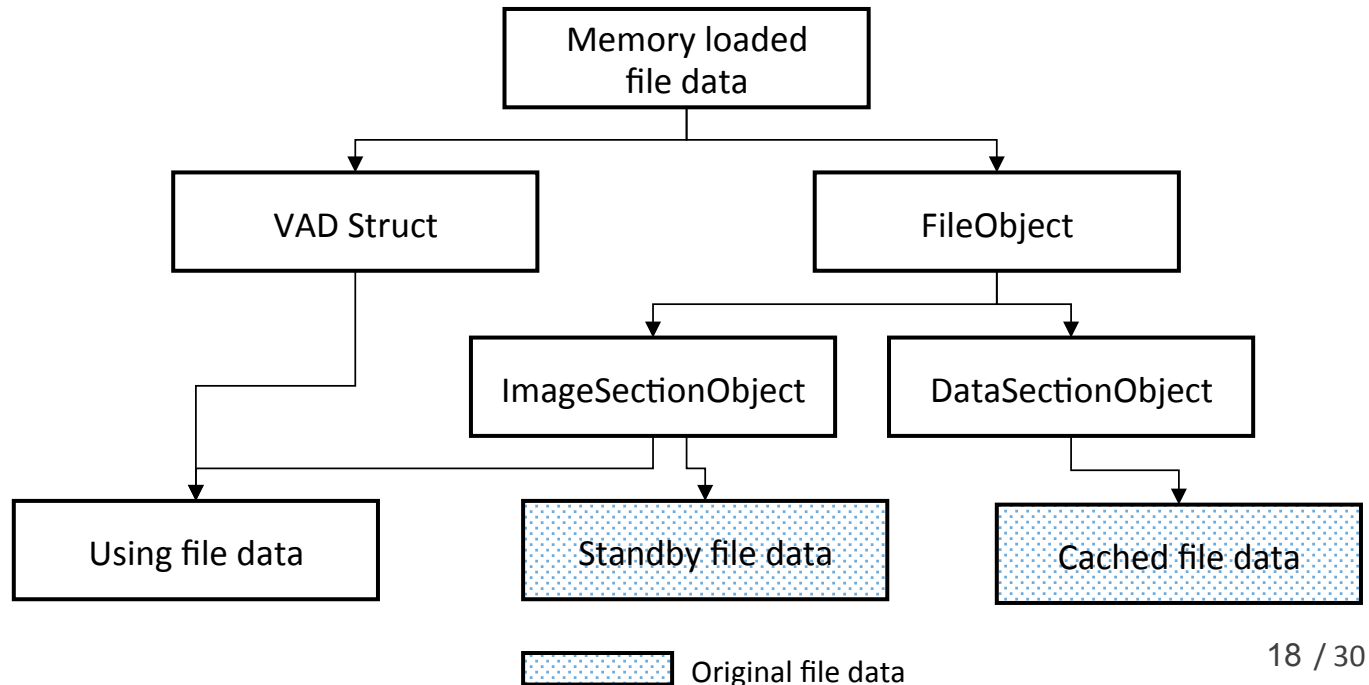
00000050 8B C9 BA D9 00 00 FF 15 6B 17 06 00 41 3B C5
00000051 0F 84 89 E3 02 48 88 00 88 22 07 00 FF 15 75
00000052 19 06 00 4C 8D 06 22 07 00 4C 8D 06 27 07
00000053 00 48 8D 15 4C 81 03 00 48 8D 00 29 20 06 00 88
00000054 75 95 01 00 4B 0A 00 00 01 00 00 01 80 44 28
00000055 00000020 D0 37 07 00 0F 85 83 E3 02 00 33 C9 FF 15 45
00000056 12 06 00 48 89 C8 E9 45 99 01 00 48 8D 05 96 0B
00000057 00 00 41 8F 5D 02 00 00 33 C9 44 89 BC 24 80 02
00000058 00 00 44 89 AC 24 84 02 00 00 48 89 84 24 86 02
00000059 00 00 44 89 AC 24 90 02 00 00 44 89 AC 24 84 02
0000005A 00 00 48 89 84 24 80 02 00 00 4C 89 AC 24 80 02
0000005B 00 FF 15 F0 11 06 00 E0 EA 00 00 4C 8D BC
0000005C 24 E0 00 00 00 46 88 C9 48 88 C8 88 D6 FF 15
0000005D 0C 11 06 00 00 48 88 BC 24 E0 00 00 4C 89 AC 24
0000005E 68 00 00 00 48 89 BC 24 A0 02 00 00 33 C9 FF 15
0000005F 84 11 06 00 00 48 89 84 24 E9 00 00 00 46 33 03 48
00000060 88 C8 48 88 05 FF 15 05 10 05 00 48 88 94 24 E8
00000061 00 00 00 33 C9 48 89 94 24 C8 02 00 00 BA 0D 7F
00000062 00 00 FF 15 28 16 06 00 33 C9 48 89 94 24 A8 02
00000063 00 00 FF 15 71 06 00 4C 8D BC 24 81 00 00
00000064 8D 55 83 45 85 C8 48 88 C8 E8 CE 69 00 00 48 8B
00000065 84 24 29 01 00 00 48 8D 2D 03 2C 06 00 48 8D BC
    
```

Offset : 0xB000

Memory mapped file

▪ Executable file data (1/3)

- VAD struct
 - Running process
- Fileobject
- ex) .exe, .dll (not .sys)



Memory mapped file

▪ Executable file data (2/3)

- Feature of memory page table
 - File data is not modified

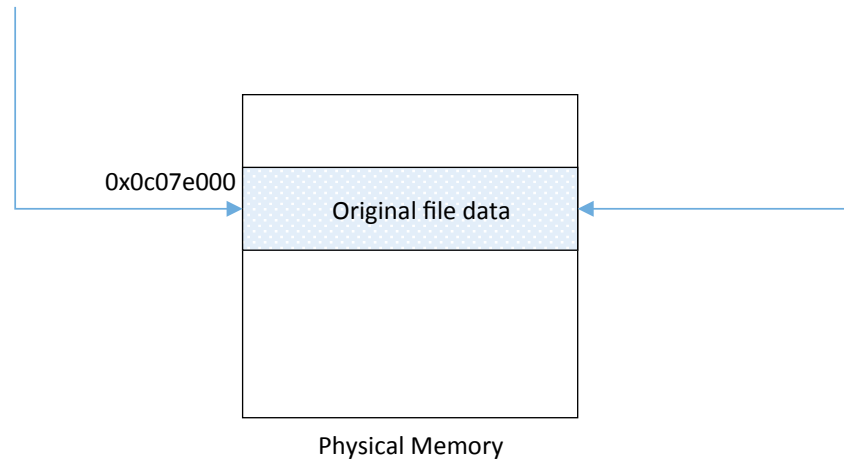
File data address : 0x0c07e000 ~

```
kd> dq fffff8a0019d0048
fffff8a0`019d0048 80000000`0c07e121 00000000`2341a121
fffff8a0`019d0058 00000000`0d81b121 00000000`0db9c121
fffff8a0`019d0068 00000000`1ed1d121 00000000`29d44860
fffff8a0`019d0078 00000000`28d9e121 00000000`2ed45860
fffff8a0`019d0088 fa800f0b`ef180460 00000000`136c6860
fffff8a0`019d0098 00000000`05b47860 00000000`0e448860
fffff8a0`019d00a8 00000000`27f1f121 00000000`19e3b860
fffff8a0`019d00b8 00000000`01d20121 00000000`28fbc860
```

ImageSectionObject -> Memory page table

```
kd> dq /p 383fc000
00000000`383fc000 e2400000`0c07e025 26900000`2341a025
00000000`383fc010 28700000`0d81b025 1fd00000`0db9c025
00000000`383fc020 10300000`1ed1d025 f8a0019d`00700400
00000000`383fc030 3bd00000`28d9e025 f8a0019d`00800400
00000000`383fc040 00000000`00000000 f8a0019d`00900400
00000000`383fc050 f8a0019d`00980400 f8a0019d`00a00400
00000000`383fc060 4aa00000`27f1f025 00000000`00000000
00000000`383fc070 35000000`01d20025 00000000`00000000
```

VAD -> Memory page table



Memory mapped file

▪ Executable file data (3/3)

- Feature of memory page table
 - File data is modified (copy-on-write)

File data address : 0x0c07e000 ~

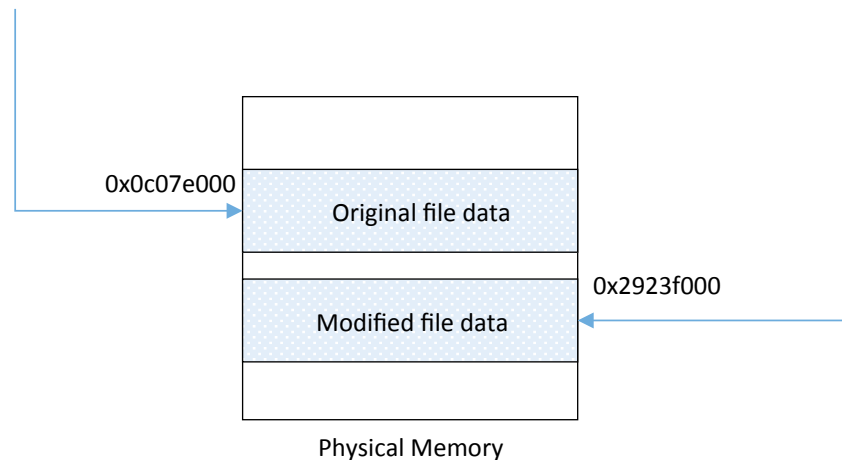
```
kd> dq fffff8a0019d0048
fffff8a0`019d0048 80000000`0c07e820 00000000`2341a121
fffff8a0`019d0058 00000000`0d81b121 00000000`0db9c121
fffff8a0`019d0068 00000000`1ed1d121 00000000`29d44860
fffff8a0`019d0078 00000000`28d9e121 00000000`2ed45860
fffff8a0`019d0088 fa800f0b`ef180460 00000000`136c6860
fffff8a0`019d0098 00000000`05b47860 00000000`0e448860
fffff8a0`019d00a8 00000000`27f1f121 00000000`19e3b860
fffff8a0`019d00b8 00000000`01d20121 00000000`28fbc860
```

ImageSectionObject -> Memory page table

File data address : 0x2923f000 ~

```
kd> dq /p 383fc000
00000000`383fc000 e2400000`2923f025 26900000`2341a025
00000000`383fc010 28700000`0d81b025 1fd00000`0db9c025
00000000`383fc020 10300000`1ed1d025 f8a0019d`00700400
00000000`383fc030 3bd00000`28d9e025 f8a0019d`00800400
00000000`383fc040 00000000`00000000 f8a0019d`00900400
00000000`383fc050 f8a0019d`00980400 f8a0019d`00a00400
00000000`383fc060 4aa00000`27f1f025 00000000`00000000
00000000`383fc070 35000000`01d20025 00000000`00000000
```

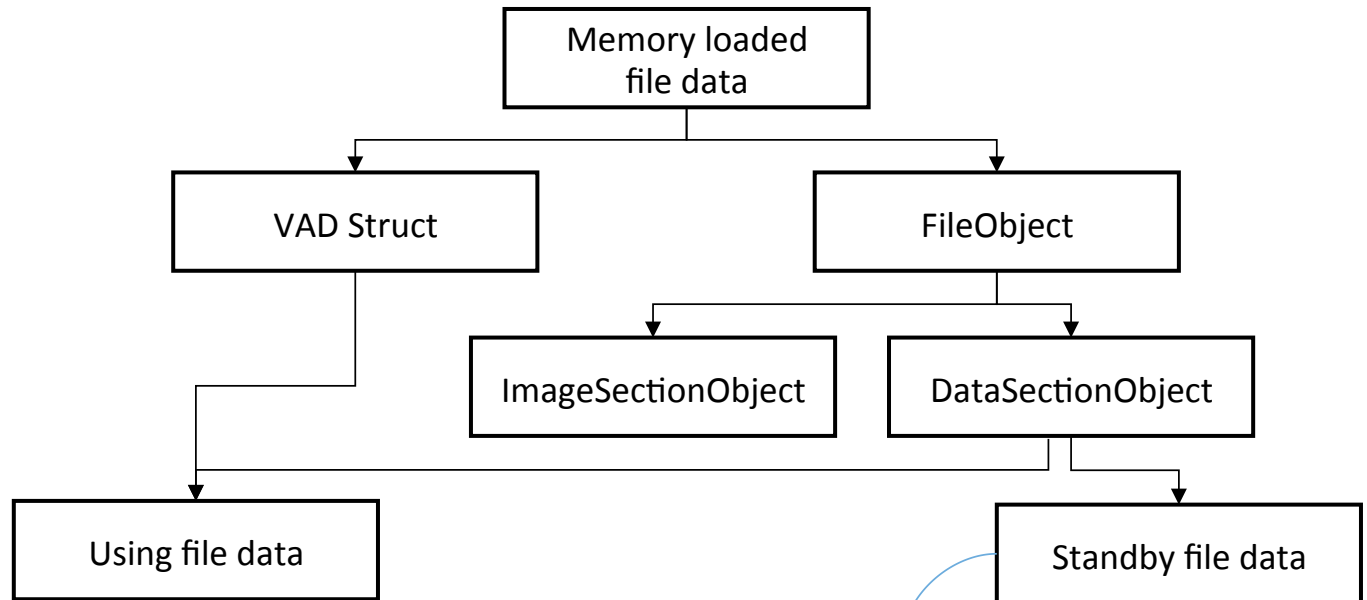
VAD -> Memory page table



Memory mapped file

▪ Normal file data (1/3)

- VAD struct
 - Running process
- Fileobject
- ex) .jpg .pdf .png ...



Original file ? -> unsure ...

Memory mapped file

■ Normal file data (2/3)

- Feature of memory page table
 - File data is not modified / File data is modified

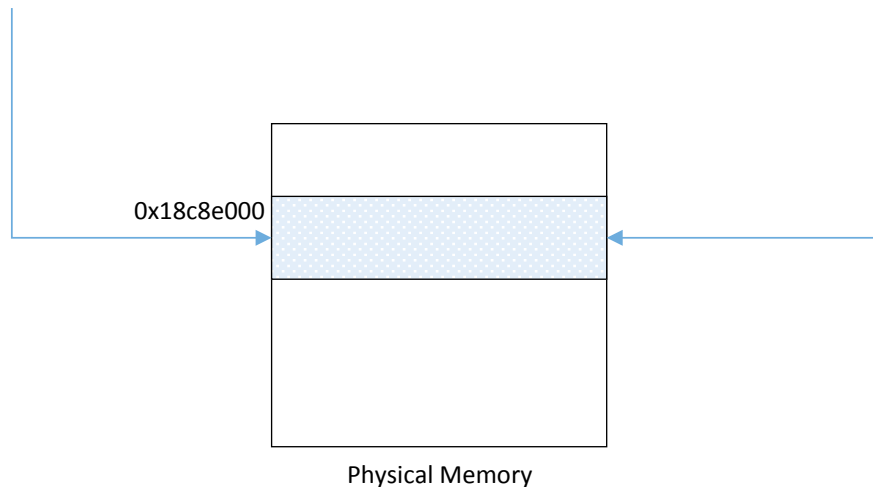
File data address : 0x18c8e000 ~

```
kd> dq fffff8a007b612c0
fffff8a0`07b612c0 00000000`18c8e963 fffffa80`0ccad150
fffff8a0`07b612d0 74536d4d`03070002 fffffa80`0ccad1b0
fffff8a0`07b612e0 fffffa80`0eec2420 00080000`00000002
fffff8a0`07b612f0 00000000`00000002 00000000`00002000
fffff8a0`07b61300 00000000`00000000 00000000`00000000
fffff8a0`07b61310 fffffa80`0d986060 00000000`03760000
fffff8a0`07b61320 fffff8a0`07b61328 80000000`1b5fa880
fffff8a0`07b61330 80000000`1b67a880 00fcfcfc`00fcfcfc
```

VAD -> Memory page table

```
kd> dq /p 10b1c80
00000000`010b1c80 acd00000`18c8e825 ffffffff`00000480
00000000`010b1c90 ffffffff`00000480 00000000`00000000
00000000`010b1ca0 00000000`00000000 00000000`00000000
00000000`010b1cb0 00000000`00000000 00000000`00000000
00000000`010b1cc0 00000000`00000000 00000000`00000000
00000000`010b1cd0 00000000`00000000 00000000`00000000
00000000`010b1ce0 00000000`00000000 00000000`00000000
00000000`010b1cf0 00000000`00000000 00000000`00000000
```

DataSectionObject -> Memory page table



Memory mapped file

■ Normal file data (3/3)

- Feature of memory page table
 - Checkout file data modified~

The diagram illustrates how memory page table entries (VAD) are mapped to hardware page table entries (ntdll!_HARDWARE_PTE). It shows two scenarios: original file data and modified file data.

Original file data:

```

kd> dq /p 10b1c80
00000000`010b1c80  acd00000`18c8e825 b3600000`19631805
00000000`010b1c90  ffffffff`00000480 00000000`00000000
00000000`010b1ca0  00000000`00000000 00000000`00000000
00000000`010b1cb0  00000000`00000000 00000000`00000000
00000000`010b1cc0  00000000`00000000 00000000`00000000
00000000`010b1cd0  00000000`00000000 00000000`00000000
00000000`010b1ce0  00000000`00000000 00000000`00000000
00000000`010b1cf0  00000000`00000000 00000000`00000000
    
```

VAD -> Memory page table

```

ntdll!_HARDWARE_PTE
+0x000 Valid           : 0y1
+0x000 Write           : 0y0
+0x000 Owner           : 0y1
+0x000 WriteThrough    : 0y0
+0x000 CacheDisable    : 0y0
+0x000 Accessed        : 0y1
+0x000 Dirty           : 0y0  Bit Pos 6 - 1Bit
+0x000 LargePage       : 0y0
+0x000 Global          : 0y0
+0x000 CopyOnWrite     : 0y0
+0x000 Prototype       : 0y0
+0x000 reserved0       : 0y1
+0x000 PageFrameNumber : 0y00000000000011000110010001110 (0x18c8e)
+0x000 reserved1       : 0y000000000000 (0)
+0x000 SoftwareWsIndex : 0y01011001101 (0x2cd)
+0x000 NoExecute       : 0y1
    
```

Original file data

Modified | file data:

```

kd> dq /p 10b1c80
00000000`010b1c80  acd00000`18c8e867 b3600000`19631825
00000000`010b1c90  c6d00000`30132825 c6e00000`1eab3825
00000000`010b1ca0  c6f00000`0081d825 c7000000`3ab1e825
00000000`010b1cb0  c7100000`1961f825 c7200000`38820825
00000000`010b1cc0  c7300000`32da1825 c7400000`03422825
00000000`010b1cd0  c7500000`16da3825 c7600000`01b24825
00000000`010b1ce0  c7700000`01826825 c7800000`27ca7825
00000000`010b1cf0  c7900000`2d1a8825 c7a00000`20729825
    
```

VAD -> Memory page table

```

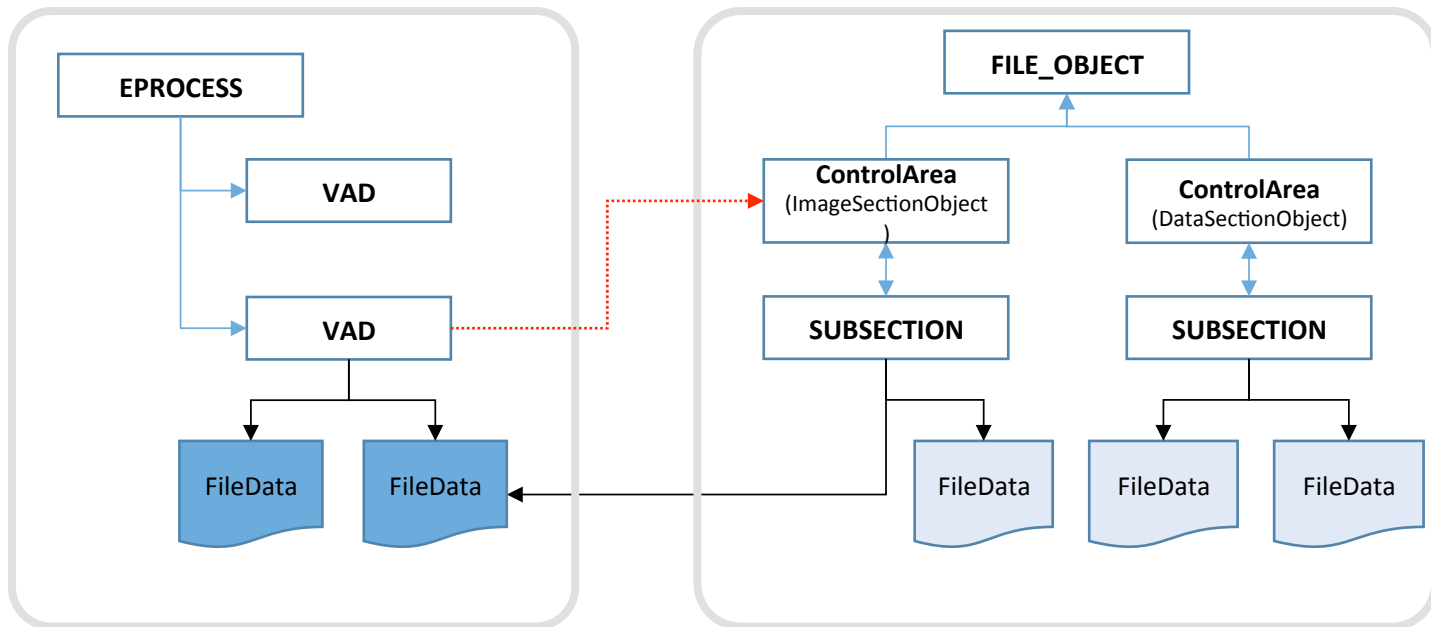
ntdll!_HARDWARE_PTE
+0x000 Valid           : 0y1
+0x000 Write           : 0y1
+0x000 Owner           : 0y1
+0x000 WriteThrough    : 0y0
+0x000 CacheDisable    : 0y0
+0x000 Accessed        : 0y1
+0x000 Dirty           : 0y1  Bit Pos 6 - 1Bit
+0x000 LargePage       : 0y0
+0x000 Global          : 0y0
+0x000 CopyOnWrite     : 0y0
+0x000 Prototype       : 0y0
+0x000 reserved0       : 0y1
+0x000 PageFrameNumber : 0y00000000000011000110010001110 (0x18c8e)
+0x000 reserved1       : 0y000000000000 (0)
+0x000 SoftwareWsIndex : 0y01011001101 (0x2cd)
+0x000 NoExecute       : 0y1
    
```

Modified | file data

Memory mapped file

■ Management of memory loaded file data

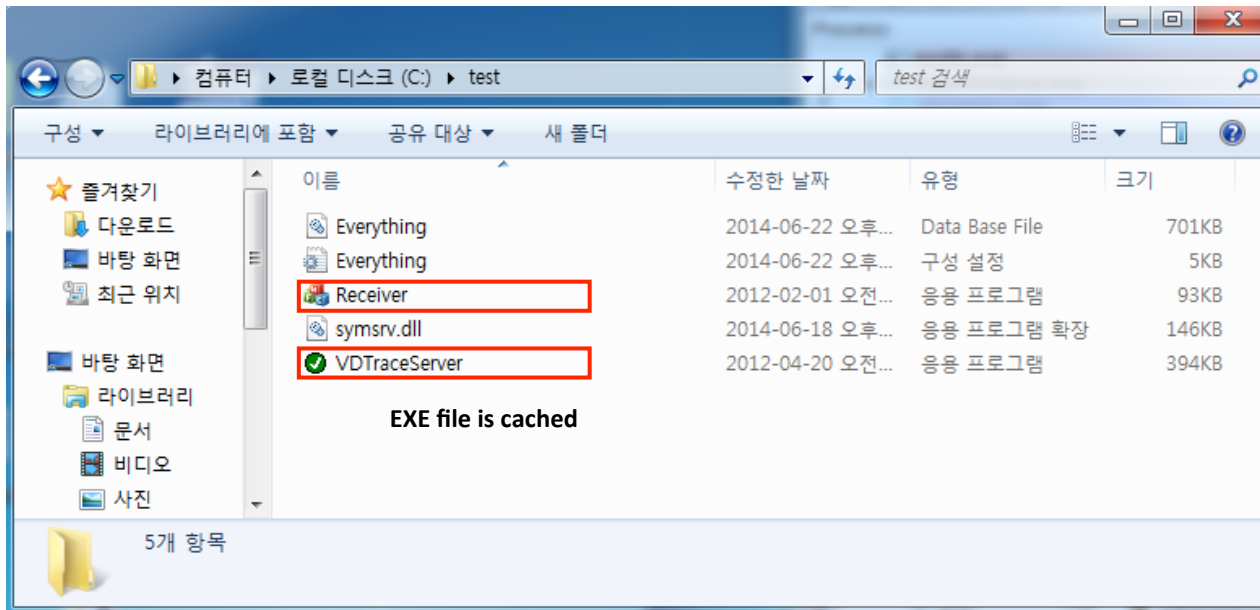
- VAD
- FILE_OBJECT



File cache

■ Feature of window file cache

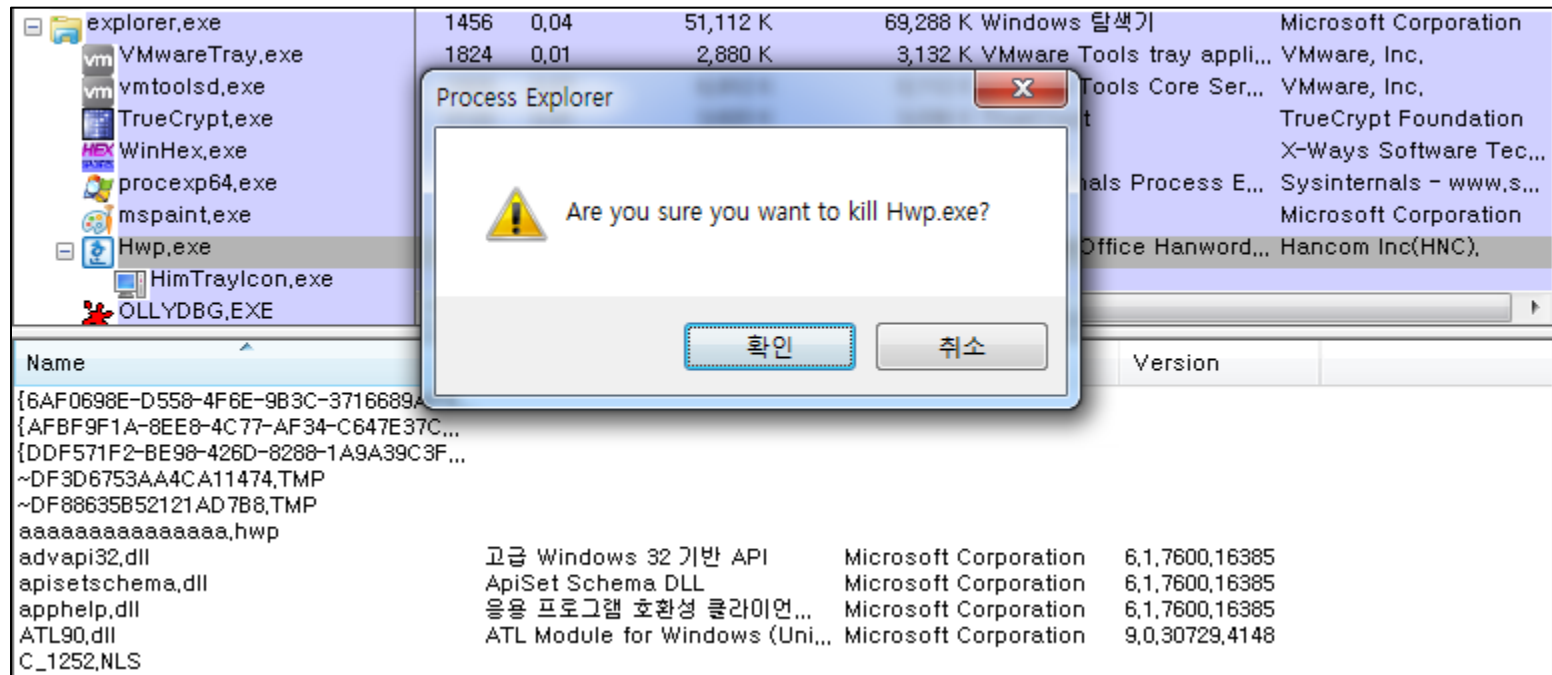
- Access Folder -> *.exe file is cached by memory management



File cache

■ Feature of window file cache

- Process End
 - Process file(.exe) is cached by memory management
 - MMF(memory mapped file) is cached by memory management



File cache

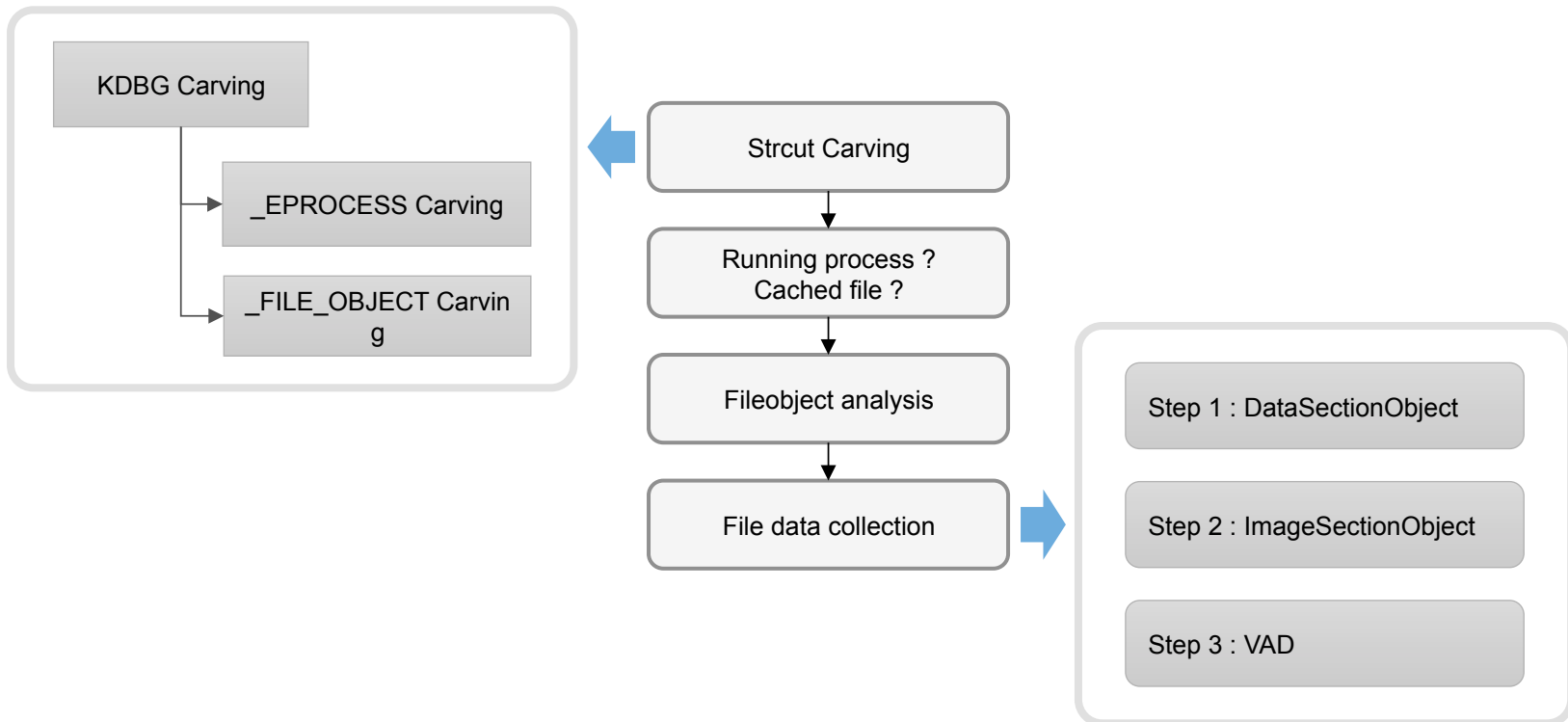
▪ How to find cached file in memory dump ?

- Carving `_FILE_OBJCET` struct
 - PoolTag = 0xE56c6946
 - PoolIndex = 0
 - `OBJECT_HEADER.TypeIndex == TYPE_FILE (0x28)`
- Check VAD
 - Running Process file data
 - `VAD -> _SUBSECTION -> _CONTROL_AREA -> _FILE_OBJECT`

Procedure for file data extraction

■ Procedure

- Kernel struct carving
- Extract file information
- File data extraction

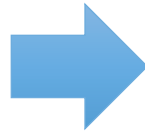


Demo

- It's demo time!



Physical Memory Image or file(hiberfil.sys)



Tool



File Extraction

Questions ?

