

파일시스템 포렌식



JK Kim

@pr0neer

forensic-proof.com

proneer@gmail.com

1. 파일 복구
2. 타임라인 분석
3. 파일/폴더 흔적 분석
4. 은닉 데이터 분석
5. 로그 분석
6. 파일시스템 터널링
7. 파일시스템 갱신 지연 시간

파일 복구

삭제된 파일 복구

- 최근 삭제된 파일이나 사건이 발생한 시점에 삭제된 파일은 **우선 분석 대상**
- 보통 파일 삭제 시 실제 데이터의 변경 없이 **파일의 메타데이터만 수정**
- 삭제된 파일의 메타데이터가 덮어써지지 않았다면 **거의 완벽한 파일 복구 가능**
 - **FAT** : 비연속적으로 할당된 파일은 앞의 연속적인 부분만 복구 가능 (FAT 테이블 초기화)
 - **NTFS** : 비연속적으로 할당된 파일도 완벽하게 복구 가능

삭제된 파일 복구

- 파일시스템 추상적 구조



- 각 파일시스템의 메타데이터

- FAT12/16/32

- ✓ FAT 영역 + 디렉터리 엔트리

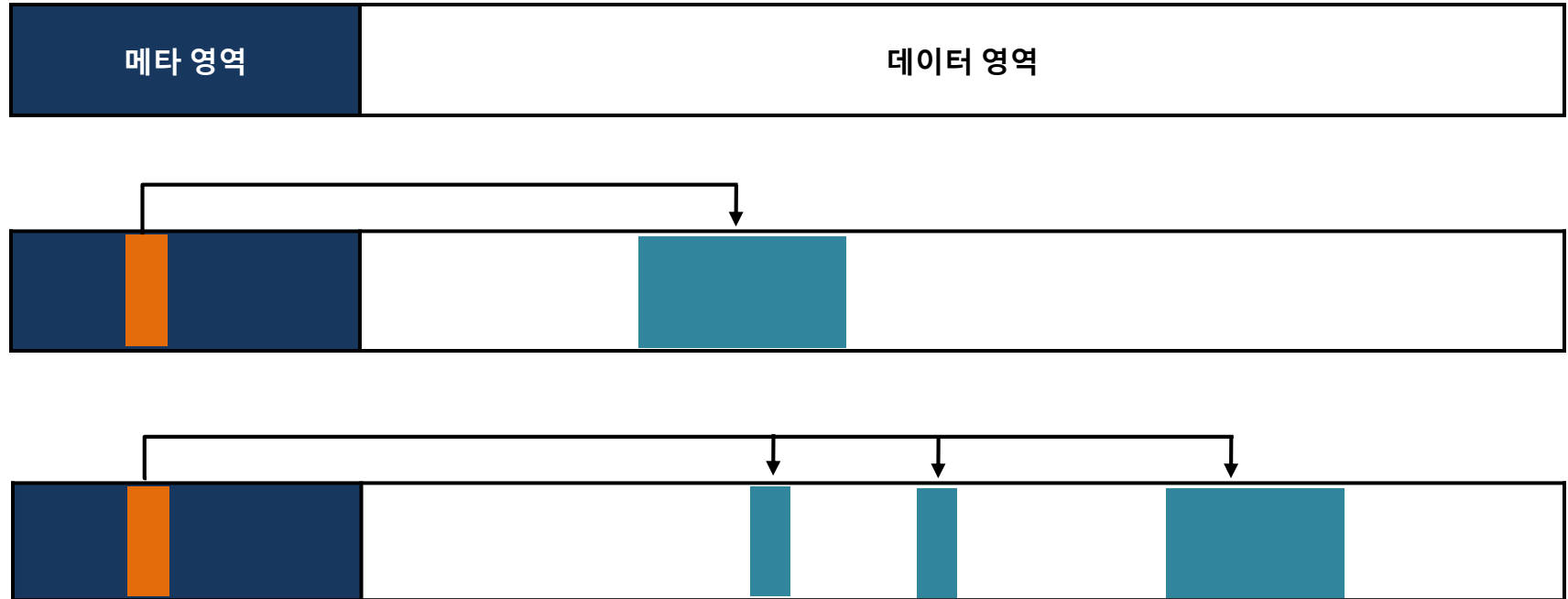
- exFAT

- ✓ 파일 디렉터리 엔트리 + 스트림 확장 엔트리 + 파일 이름 확장 엔트리

- NTFS

- ✓ MFT 레코드 (\$STD_INFO, \$FNA, \$DATA 속성 등)

삭제된 파일 복구



- 파일의 메타데이터가 존재한다면 100% 복구
- 운영체제 별 혹은 시스템 볼륨 여부에 따라 차이가 있음
- 파일 삭제 시 메타데이터가 아닌 실제 데이터가 덮어써질 가능성은?

삭제된 파일 복구

▪ FAT의 삭제된 파일 탐색

- 루트 디렉터리부터 하위 디렉터리까지 디렉터리 엔트리 탐색
- 디렉터리 엔트리의 첫 바이트가 "0xE5"인 엔트리 수집

▪ NTFS의 삭제된 파일 탐색

- \$MFT의 \$BITMAP 속성에서 0x00값을 가지는 MFT 레코드 조사
- MFT 레코드 헤더의 플래그 값이 0x00인 MFT 레코드 수집

데이터 카빙

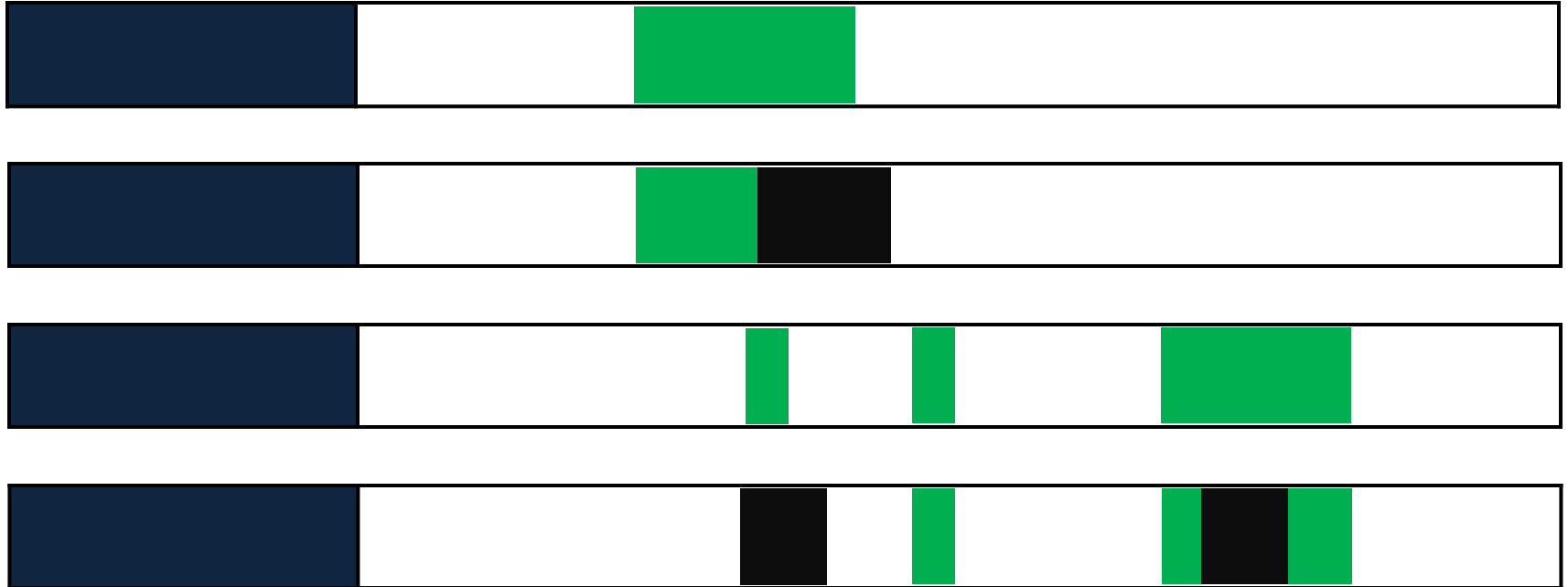
- 파일 메타데이터가 없는 경우, 파일 형식만을 가지고 복구
- 볼륨의 비할당 영역이나 메모리 등과 같은 비트스트림에 적용
- N-GRAM의 윈도우 크기는?

111001110110011100110100110100101110001111111110
111001110110011100110100110100101110001111111110
1110011101100111001100110100110100101110001111111110

Sliding window (window size = 3) → 3-GRAM

- 이전 볼륨의 복구 여부에 따라 결정
- 현재 볼륨의 파일만 복구한다면 윈도우 크기는 "클러스터 크기"

데이터 카빙

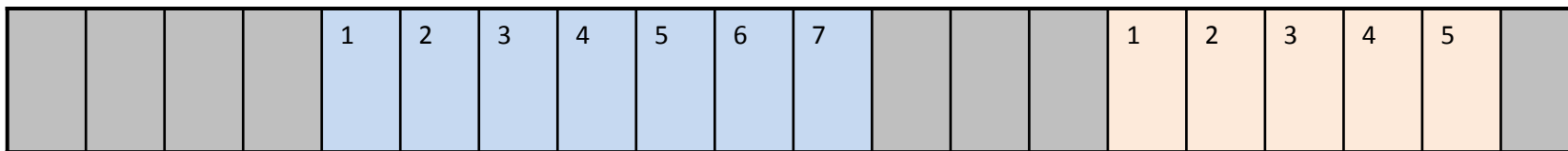


- 메타데이터가 없는 경우 개별 파일 구조에 기반하여 복구
- 파일 카빙은 비트스트림에서 의미 있는 정보를 획득하는 기법
- 그래픽 이미지, 문서, 실행파일, 데이터 구조, 문자열 등

데이터 카빙

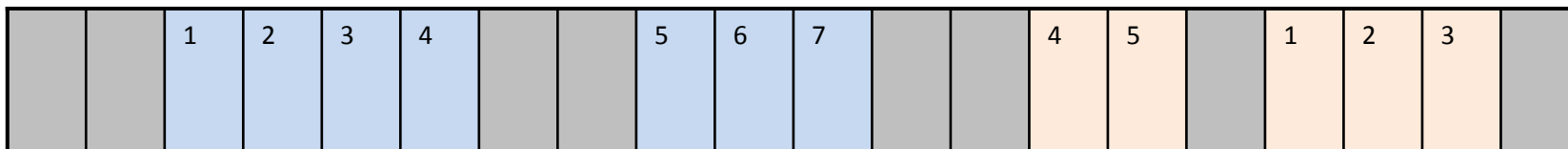
■ 연속적인 카빙

- 데이터가 볼륨의 연속된 공간에 저장된 경우 복구하는 기법



■ 비연속적인 카빙

- 데이터의 단편화가 발생하여 여러 부분으로 조각나 볼륨에 저장되는 경우 복구하는 기법



타임라인 분석

파일시스템 타임라인 분석

- 파일시스템의 시간 정보를 기반으로 분석하는 기법
- 시간 필드를 기준으로 정렬하는 방법
 - 생성 시간
 - 수정 시간
 - 접근 시간
 - (NTFS) 엔트리 수정 시간
- 시간 필드를 통합하는 방법
 - 모든 시간 정보를 시간을 기준으로 단일 필드로 정렬

파일시스템 시간 표현 형식

- MACB vs. MACE

NTFS 시간	MACB	MACE
생성 시간 (Created)	B (Birth)	C (Creation)
수정 시간 (Last Modified)	M (Modification)	M (Modification)
접근 시간 (Last Accessed)	A (Access)	A (Access)
엔트리 수정 시간 (Entry Modified)	C (Change)	E (Entry Modified)

파일시스템 시간 정보

- **FAT**

- **디렉터리 엔트리** – DOS 시간 형식, 2초 카운팅
 - ✓ 생성 날짜/시간(+10분의 1초), 수정 날짜/시간, 접근 날짜

- **exFAT**

- **파일 디렉터리 엔트리** – DOS 시간 형식, 타임존 오프셋 지원
 - ✓ 생성 날짜/시간, 수정 날짜/시간, 접근 날짜/시간

- **NTFS**

- **\$STD_INFO, \$FNA** – 윈도우 64비트 시간 형식, 100 나노초 카운팅
 - ✓ 생성 날짜/시간, 수정 날짜/시간, 접근 날짜/시간, MFT 레코드 수정 날짜/시간

타임라인 분석

파일 및 폴더 행위 분석

■ NTFS

- 총 8개의 시간 정보(\$SIA 4개, \$FNA 4개)
- 각 시간 정보를 잘!!! 분석하면 파일 및 폴더의 행위 흔적 분석 가능

\$STANDARD_INFO	생성	수정	접근	삭제	복사	로컬이동	볼륨이동	이름변경
Created	✓				✓			
Written	✓	✓						
Accessed	✓	✓	✓		✓		✓	
MFT Modified	✓			✓	✓	✓	✓	✓

\$FILE_NAME	생성	수정	접근	삭제	복사	로컬이동	볼륨이동	이름변경
Created	✓				✓		✓	
Written	✓			✓	✓	✓	✓	
Accessed	✓				✓		✓	
MFT Modified	✓			✓	✓	✓	✓	

파일시스템 시간 조작 분석

▪ 시간 조작

- 악성코드는 자신을 은닉하기 위해 시스템 주요 파일(ntdll.dll, rundll32.exe 등)과 시간 동기화
- **SetFileTime() API (Kernel32.dll)**
 - ✓ 생성, 수정, 접근 시간만 수정 가능
 - ✓ MFT 레코드 수정 시간을 이용해 쉽게 탐지 가능
- **NtSetInformationFile() API (NTDLL.dll)**
 - ✓ 생성, 수정, 접근, MFT 레코드 수정 시간 모두 변경 가능
 - ✓ \$FILE_NAME 속성을 이용해 탐지 가능
- **\$STANDARD_INFORMATION, \$FILE_NAME의 8개 시간을 모두 수정한 경우는?**

타임라인 분석

파일시스템 시간 조작 분석

■ 시간 조작 탐지

- 생성 시간, MFT 레코드 수정 시간 등을 정렬하여 시스템 설치 시간 대에 의심 파일 생성 확인

Drive C: Windows\System32 21 days ago

Name	Ext.	Size	Created ^	Modified	Accessed	Record update	Attr.	1st sector
PSHED.DLL	DLL	56.1 KB	2009-07-14 08:19:28	2009-07-14 10:45:45	2009-07-14 08:19:28	2012-03-04 10:07:36	A	185792
clfs32.dll	dll	77.5 KB	2009-07-14 08:19:34	2009-07-14 10:40:15	2009-07-14 08:19:34	2012-03-04 10:06:59	A	139017...
txfw32.dll	dll	11.5 KB	2009-07-14 08:19:38	2009-07-14 10:41:55	2009-07-14 08:19:38	2012-03-04 10:07:44	A	8168856
services.exe	exe	321 KB	2009-07-14 08:19:46	2009-07-14 10:39:37	2009-07-14 08:19:46	2012-03-04 10:07:39	A	226920
csrss.exe	exe	7.5 KB	2009-07-14 08:19:49	2009-07-14 10:39:02	2009-07-14 08:19:49	2012-03-04 10:07:00	A	119760
smss.exe	exe	110 KB	2009-07-14 08:19:50	2009-07-14 10:39:41	2009-07-14 08:19:50	2012-03-04 10:07:41	A	436696
clfs.sys	sys	359 KB	2009-07-14 08:19:59	2009-07-14 10:52:31	2009-07-14 08:19:59	2012-03-04 10:06:59	A	367256
api-ms-win-security-lsal...	dll	3.5 KB	2009-07-14 08:20:47	2009-07-14 10:24:53	2009-07-14 08:20:47	2012-03-04 10:06:55	HA	110636...
api-ms-win-security-sdd...	dll	3.0 KB	2009-07-14 08:20:47	2009-07-14 10:24:53	2009-07-14 08:20:47	2012-03-04 10:06:55	HA	110666...
sechost.dll	dll	111 KB	2009-07-14 08:20:52	2009-07-14 10:41:53	2009-07-14 08:20:52	2012-03-04 10:07:39	A	332464
cryptbase.dll	dll	43.0 KB	2009-07-14 08:20:54	2009-07-14 10:40:24	2009-07-14 08:20:54	2012-03-04 10:07:00	A	94720
profapi.dll	dll	43.0 KB	2009-07-14 08:20:57	2009-07-14 10:41:53	2009-07-14 08:20:57	2012-03-04 10:07:36	A	134208
netevent.dll	dll	18.5 KB	2009-07-14 08:20:58	2009-07-14 10:30:47	2009-07-14 08:20:58	2012-03-04 10:07:19	A	144538...
nsi.dll	dll	13.5 KB	2009-07-14 08:21:05	2009-07-14 10:41:53	2009-07-14 08:21:05	2012-03-04 10:07:33	A	103296
RpcEpMap.dll	dll	65.5 KB	2009-07-14 08:21:05	2009-07-14 10:41:53	2009-07-14 08:21:05	2012-03-04 10:07:37	A	133952
winnsi.dll	dll	25.5 KB	2009-07-14 08:21:08	2009-07-14 10:41:56	2009-07-14 08:21:08	2012-03-04 10:07:48	A	176744
dhcpcsvc6.dll	dll	53.0 KB	2009-07-14 08:21:09	2009-07-14 10:40:28	2009-07-14 08:21:09	2012-03-08 08:32:33	A	192704
dhcpcsvc.dll	dll	85.0 KB	2009-07-14 08:21:09	2009-07-14 10:40:28	2009-07-14 08:21:09	2012-03-08 08:32:33	A	151104
dhcpcore6.dll	dll	219 KB	2009-07-14 08:21:13	2009-07-14 10:40:28	2009-07-14 08:21:13	2012-03-08 08:32:33	A	447512
IPHLPAPI.DLL	DLL	143 KB	2009-07-14 08:21:13	2009-07-14 10:41:10	2009-07-14 08:21:13	2012-03-04 10:07:11	A	276512
dhcpcore.dll	dll	307 KB	2009-07-14 08:21:15	2009-07-14 10:40:28	2009-07-14 08:21:15	2012-03-04 10:07:02	A	446896
api-ms-win-core-ums-l1...	dll	3.0 KB	2009-07-14 08:21:15	2009-07-14 10:24:53	2009-07-14 08:21:15	2012-03-04 10:06:55	HA	110257...
shimeng.dll	dll	6.5 KB	2009-07-14 08:21:19	2009-07-14 10:41:54	2009-07-14 08:21:19	2012-03-08 08:32:37	A	8033728

파일시스템 시간 조작 분석

- 시간 조작 도구

- setMACE

- ✓ <http://reboot.pro/files/file/91-setmace/>

- TimeStomp

- ✓ <http://www.offensive-security.com/metasploit-unleashed/Timestomp>

타임라인 분석

실습 #1

- WinHex를 이용해 로컬 볼륨 타임라인 분석하기!!

파일/폴더 흔적 분석

악성코드 선호 경로

- **흔하지 않은 경로에 파일 생성**
 - AV 실시간 탐지를 우회하기 위한 목적
 - 사용자에게 인지되지 않고 장기간 은닉하기 위한 목적
- **분석 방법**
 - 선호 경로에 위치한 실행 파일을 수집하여 분석
 - 특정 경로에 실행 파일이 위치할 경우 악성코드 가능성 99%

악성코드 선호 경로

■ 윈도우 주요 폴더

- %SystemRoot%
- %SystemRoot%\system%
- %SystemRoot%\System32%
- %SystemRoot%\System32%\dllcache%
- %SystemRoot%\System32%\drivers%

■ WoW64

- %SystemRoot%\SysWOW64%
- %SystemRoot%\SysWOW64%\dllcache%
- %SystemRoot%\SysWOW64%\drivers%

악성코드 선호 경로

▪ 사용자 프로필 폴더

- %SystemDrive%\Default\
- %SystemDrive%\Public\
- %SystemDrive%\<USER>\

▪ 사용자 데이터 폴더

- %UserProfile%\AppData\
- %UserProfile%\AppData\Local\
- %UserProfile%\AppData\Roaming\

▪ 휴지통 폴더

- %SystemDrive%\\$Recycle.Bin\

악성코드 선호 경로

- 프로그램 데이터 폴더

- %SystemDrive%\ProgramData\ (%SystemDrive%\All Users\)

- 시스템 볼륨 정보 폴더

- %SystemDrive%\System Volume Information\

- 임시 폴더

- %UserProfile%\AppData\Local\Temp\
- %SystemRoot%\Temp\

- 인터넷 캐시 폴더

- %UserProfile%\AppData\Local\Microsoft\Windows\Temporary Internet Files\

악성코드 선호 경로

- **액티브X 폴더**

- %SystemRoot%\Downloaded Program Files\

- **시작 프로그램 폴더**

- %UserProfile%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\

- **작업 스케줄러 폴더**

- %SystemRoot%\Tasks\

- **알려진 폴더**

- %SystemDrive%\Intel\

비정상 파일

- **한 글자 파일명**
 - a.gif, b.jpg, g.exe, v.exe, ...
- **랜덤한 파일명**
 - hdpfoi.exe, yyr.exe, 3378.exe, 499389.exe, ...
- **확장자 변경으로 시그니처 불일치**
 - abc.jpg, gcc.gif – PE 파일
- **대체 데이터 스트림(ADS, Alternative Data Stream)**
 - C:\Windows\System32\scvhost.exe

비정상 파일

▪ “svchost.exe” 위장 악성파일명

svchost	svcehost	svchostc32	svghost
svch0st	svphost	szchostc	svchostms
svchosts	svchostdll	svehost	svchostxxx
scvhost	svvhosti	srvchost	suchostp
svhost	sach0st	svchosts32	suchosts
svohost	swchost	scvhosv	smsvchost
svchest	servehost	ssvichosst	svchost0
svchost32	svsh0st	svrhost	svchost64
suchost	svchsot	svichosst	svchöst
svshost	scchostc	svchoxt	s_host
svchast	snvhost	svchost_cz	svchost”
svcnost	scchost	schost	svphostu
syshost	svvhost	ssvchost	svchosting
svhcst	svahost	sv±hest	sachostc
svchost	svcinit	shhost	sachostw
svchon32	ssvch0st	svchostt	svshoct
svchost2	svchots	svchosf	svchpst
Svcchost	svdhost	svchostp	svohcst
sxhost	svchostv	sachostp	scanost
svchost31	scvchusts	sachosts	schosts
syschost	svchostxi	sachostx	svcroot
svchîst	st#host	swhost	svschost
synchost	svchost3	scvh0st	scvhosts

은닉 데이터 분석

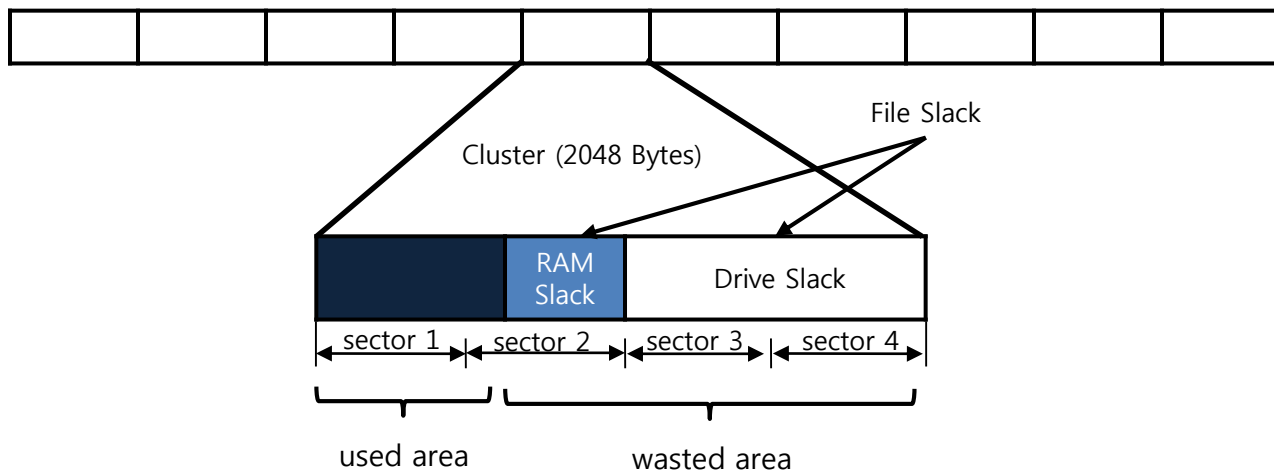
숨긴/암호화 파일

- 사용자의 의도적인 행위가 포함됐을 가능성
- 숨긴 파일 탐색
 - **FAT** – 디렉터리 엔트리의 속성 값이 0x02를 갖는 파일 탐색
 - **exFAT** – 파일 디렉터리 엔트리의 속성 값이 0x02를 갖는 파일 탐색
 - **NTFS** – \$STANDARD_INFORMATION 속성의 플래그 값이 0x0002를 갖는 파일 탐색
- 암호화된 파일 탐색
 - **NTFS** – \$STANDARD_INFORMATION 속성의 플래그 값이 0x4000을 갖는 파일 탐색
 - ✓ 레지스트리의 암호화된 private key 복호화 (무차별 대입)
 - ✓ EFS0.TMP 파일 조사

시그니처 분석

- 파일 시그니처와 확장자가 일치하는지 검사
- 윈도우는 확장자 기반의 애플리케이션 바인딩 사용
- 확장자를 변경해 파일을 은폐하거나 사용자의 클릭 유도!!
- 확장자 위치
 - FAT – 디렉터리 엔트리
 - exFAT – 파일 이름 확장 디렉터리 엔트리
 - NTFS – \$FILE_NAME 속성

슬랙 공간 분석



■ 슬랙

- 물리적인 구조와 논리적인 구조의 차이로 발생하는 낭비되는 공간
- 의도적으로 삽입한 데이터나 이전 파일의 데이터가 남아있을 가능성
- 파일 슬랙 이외에 DB 레코드 슬랙과 같이 다양한 부분 조사
- 램 슬랙, 드라이브 슬랙, 파일시스템 슬랙, 볼륨 슬랙, MBR 슬랙, MFT 슬랙, INDX 슬랙 등 조사

슬랙 공간 분석

- 파일시스템 구조 상 낭비되는 영역은 악의적인 데이터를 은폐하는데 활용
- **MBR 슬랙**
 - MBR부터 VBR 사이에 남는 공간
 - 윈도우 XP 이전 (과거 fdisk와 같은 디스크 유틸리티는 트랙 할당 방식 사용)
 - 윈도우 Vista 이후 : 2,048 섹터 (최근에는 1MiB 할당 방식 사용)
- **FAT12/16/32**
 - 예약된 영역의 낭비되는 섹터(0,1,2,6,7,8 섹터 제외) 조사
 - FSINFO 섹터(1,7 섹터)의 사용되지 않는 영역 조사
 - 추가 부트 코드 섹터(2,8 섹터) 영역 조사

슬랙 공간 분석

▪ exFAT

- VBR의 확장 부트 코드 영역(1~8 섹터)와 예약된 영역(10 섹터) 분석
- 백업 VBR 영역 조사

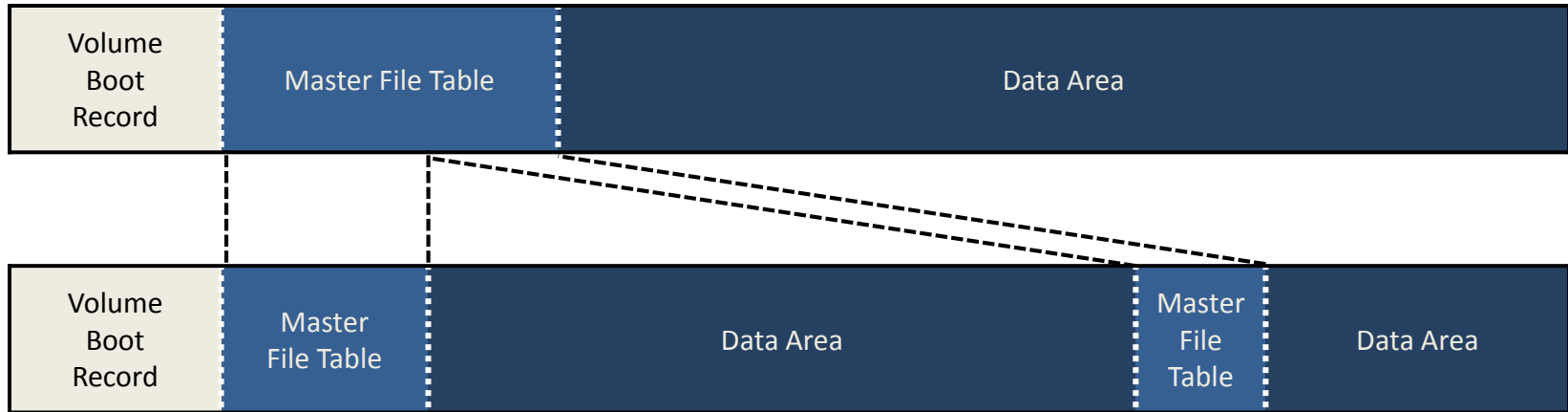
▪ NTFS

- VBR의 낭비되는 영역 조사
- MFT 레코드 12~15번 영역 조사
- 백업 VBR 영역 조사

▪ HPA(Host Protected Area), DCO(Device Configuration Overlay) 조사

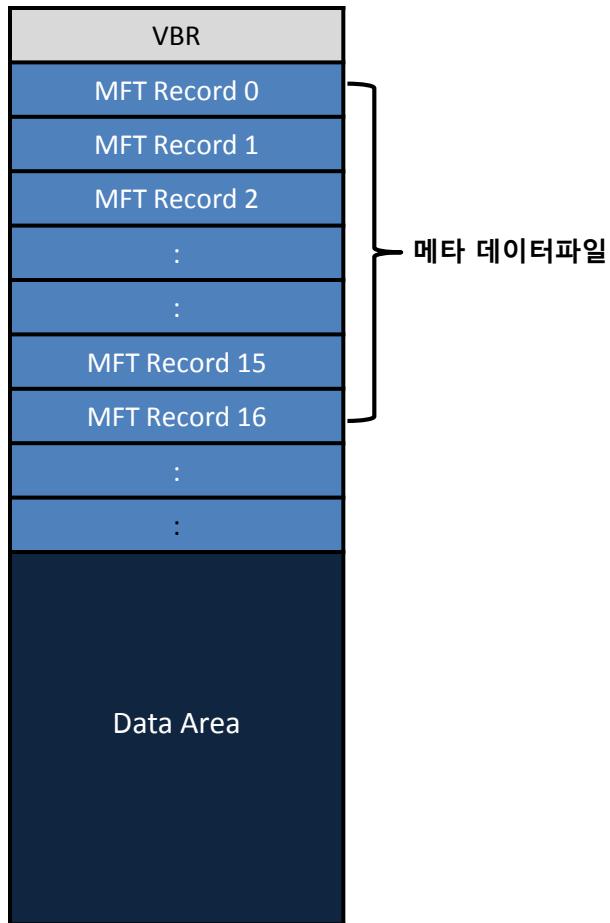
슬랙 공간 분석 - MFT, INDX 슬랙 분석

- NTFS 파일시스템 추상화 구조



슬랙 공간 분석

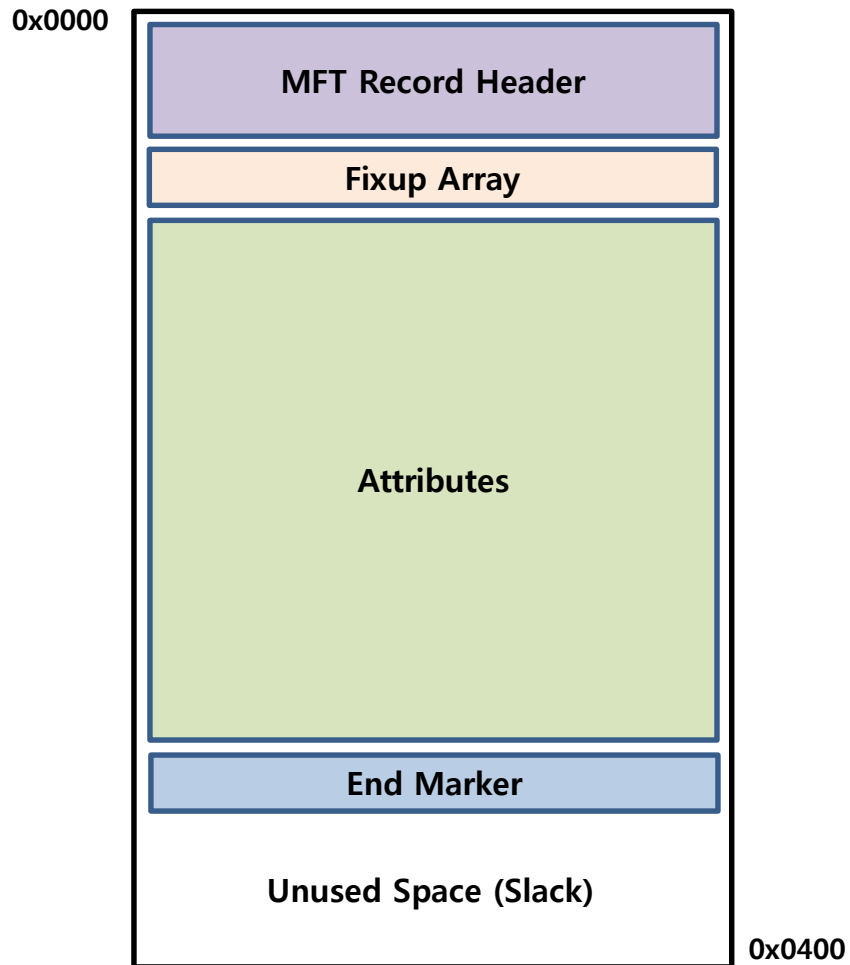
■ MFT 레코드



번호	이름	설명
0	\$MFT	MFT에 대한 MFT Entry
1	\$MFTMirr	\$MFT 파일의 일부 백업본
2	\$LogFile	메타데이터(MFT)의 트랜잭션 저널 정보
3	\$Volume	볼륨의 레이블, 식별자, 버전 등의 정보
4	\$AttrDef	속성의 식별자, 이름, 크기 등의 정보
5	.	볼륨의 루트 디렉터리
6	\$Bitmap	볼륨의 클러스터 할당 정보
7	\$Boot	볼륨이 부팅 가능할 경우 부트 섹터 정보
8	\$BadClus	배드 섹터를 가지는 클러스터 정보
9	\$Secure	파일의 보안, 접근 제어와 관련된 정보
10	\$Upcase	모든 유니코드 문자의 대문자
11	\$Extend	\$ObjID, \$Quota, \$Reparse points, \$UsnJrnl 등의 추가적인 파일의 정보를 기록하기 위해 사용
12 - 15		미래를 위해 예약
16 -		포맷 후 생성되는 파일의 정보를 위해 사용
-	\$ObjId	파일 고유의 ID 정보 (Windows 2000 -)
-	\$Quota	사용량 정보 (Windows 2000 -)
-	\$Reparse	Reparse Point 에 대한 정보 (Windows 2000 -)
-	\$UsnJrnl	파일, 디렉터리의 변경 정보 (Windows 2000 -)

슬랙 공간 분석

- MFT 레코드



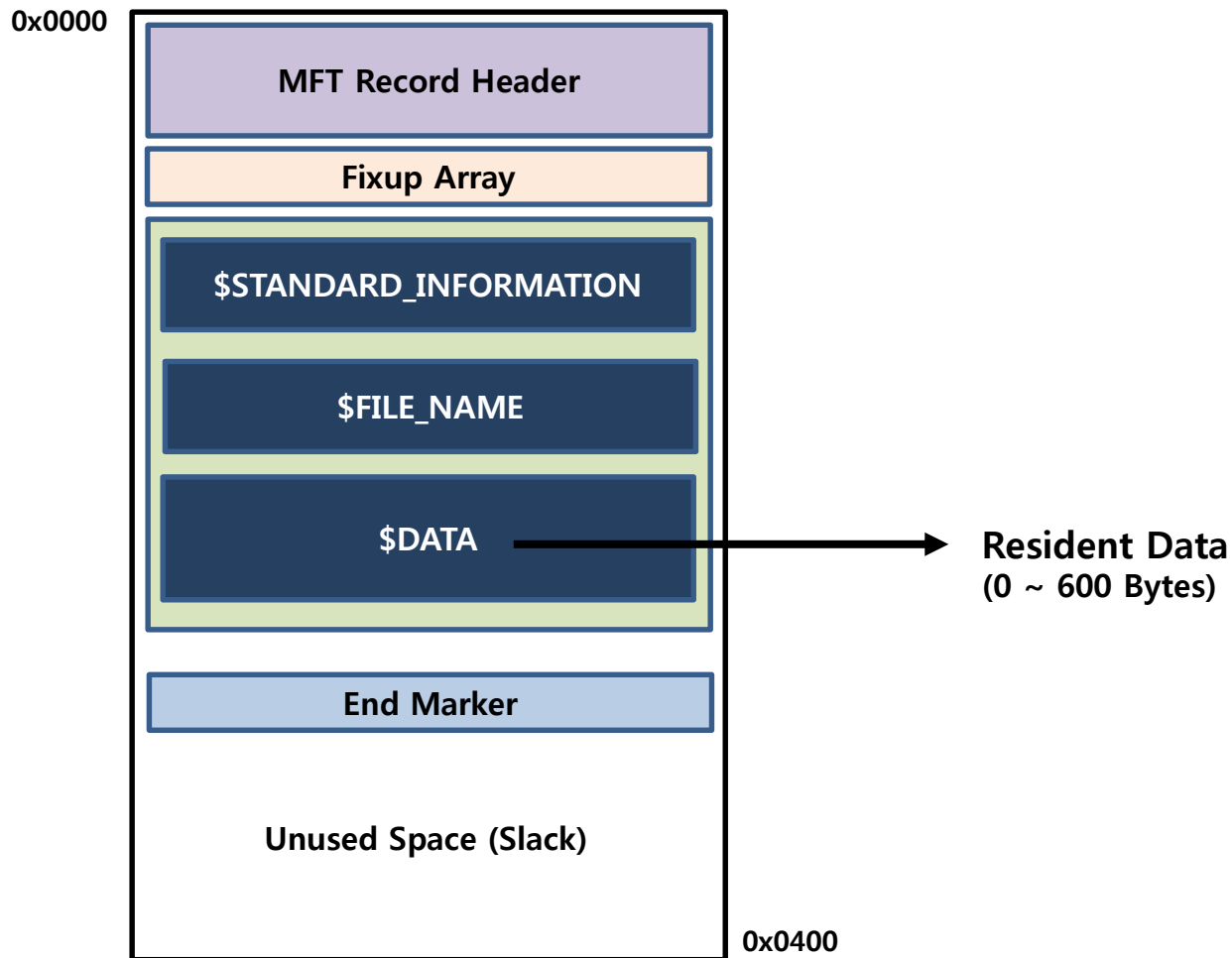
슬랙 공간 분석

▪ MFT 레코드 속성

속성 식별값		속성이름	설명
16	0x10	\$STANDARD_INFORMATION	파일의 생성.접근.수정 시간, 소유자 등의 일반적인 정보
32	0x20	\$ATTRIBUTE_LIST	추가적인 속성들의 리스트
48	0x30	\$FILE_NAME	파일 이름(유니코드), 파일의 생성.접근.수정 시간
64	0x40	\$VOLUME_VERSION	볼륨 정보 (Windows NT 1.2 버전에만 존재)
64	0x40	\$OBJECT_ID	16바이트의 파일, 디렉터리의 고유 값, 3.0 이상에서만 존재
80	0x50	\$SECURITY_DESCRIPTOR	파일의 접근 제어와 보안 속성
96	0x60	\$VOLUME_NAME	볼륨 이름
112	0x70	\$VOLUME_INFORMATION	파일 시스템의 버전과 다양한 플래그
128	0x80	\$DATA	파일 내용
144	0x90	\$INDEX_ROOT	인덱스 트리의 루트 노드
160	0xA0	\$INDEX_ALLOCATION	인덱스 트리의 루트와 연결된 노드
176	0xB0	\$BITMAP	\$MFT와 인덱스의 할당 정보 관리
192	0xC0	\$SYMBOLIC_LINK	심볼릭 링크 정보 (Windows 2000+)
192	0xC0	\$REPARSE_POINT	심볼릭 링크에서 사용하는 reparse point 정보 (Windows 2000+)
208	0xD0	\$EA_INFORMATION	OS/2 응용 프로그램과 호환성을 위해 사용 (HPFS)
224	0xE0	\$EA	OS/2 응용 프로그램과 호환성을 위해 사용 (HPFS)
256	0x100	\$LOGGED_UTILITY_STREAM	암호화된 속성의 정보와 키 값 (Windows 2000+)

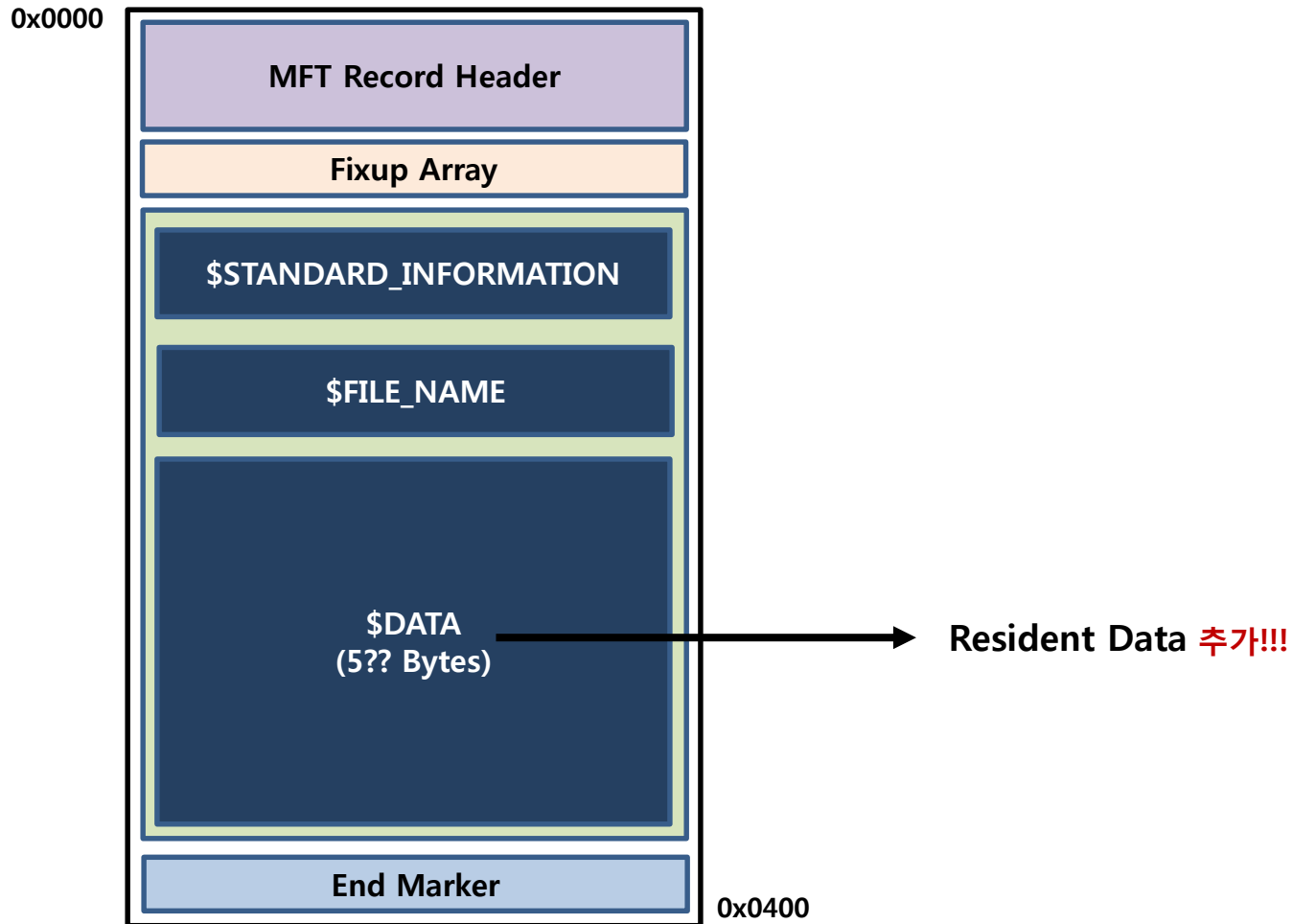
슬랙 공간 분석

- MFT 레코드 구조 ➔ 일반 파일



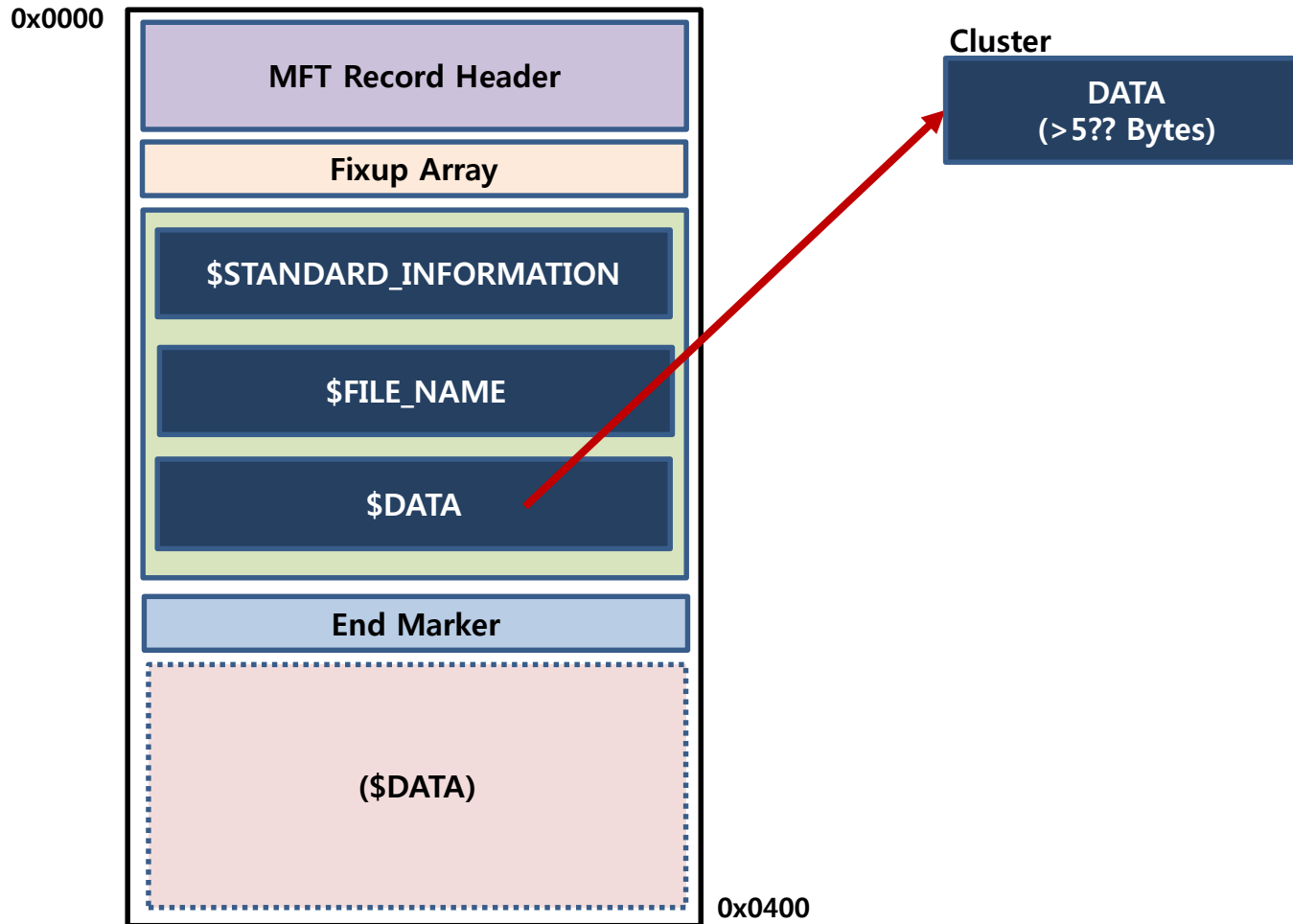
슬랙 공간 분석

- MFT 레코드 구조 ➔ 일반 파일



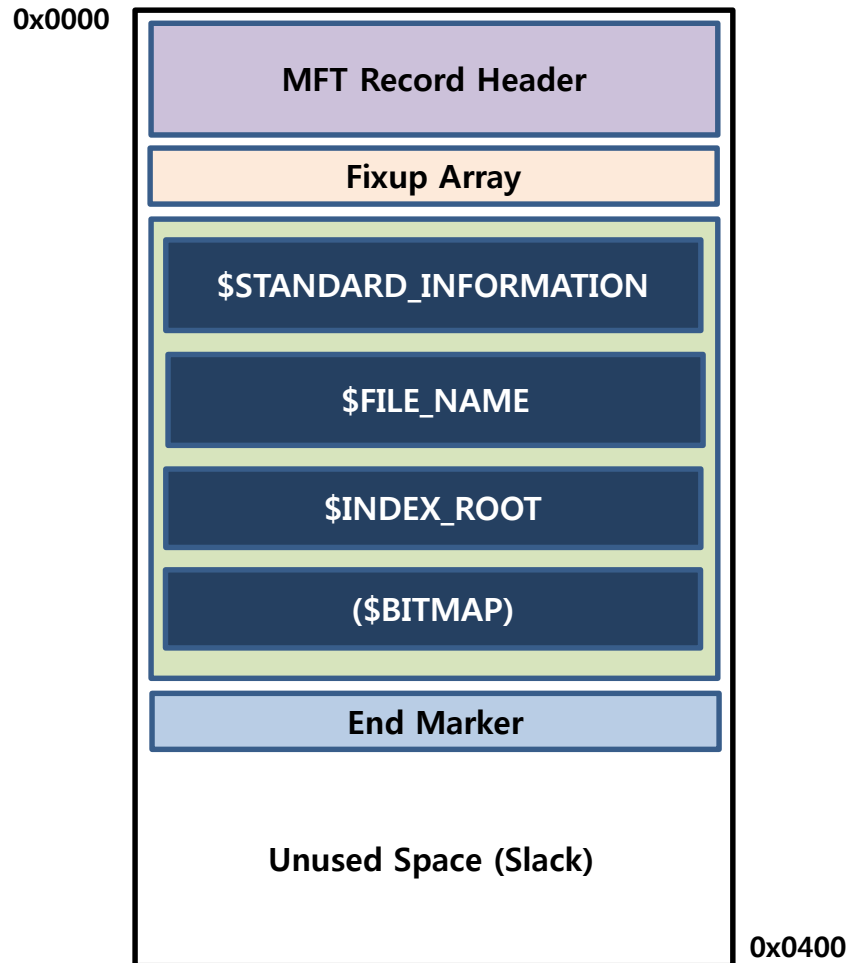
슬랙 공간 분석

- MFT 레코드 구조 ➔ 일반 파일



슬랙 공간 분석

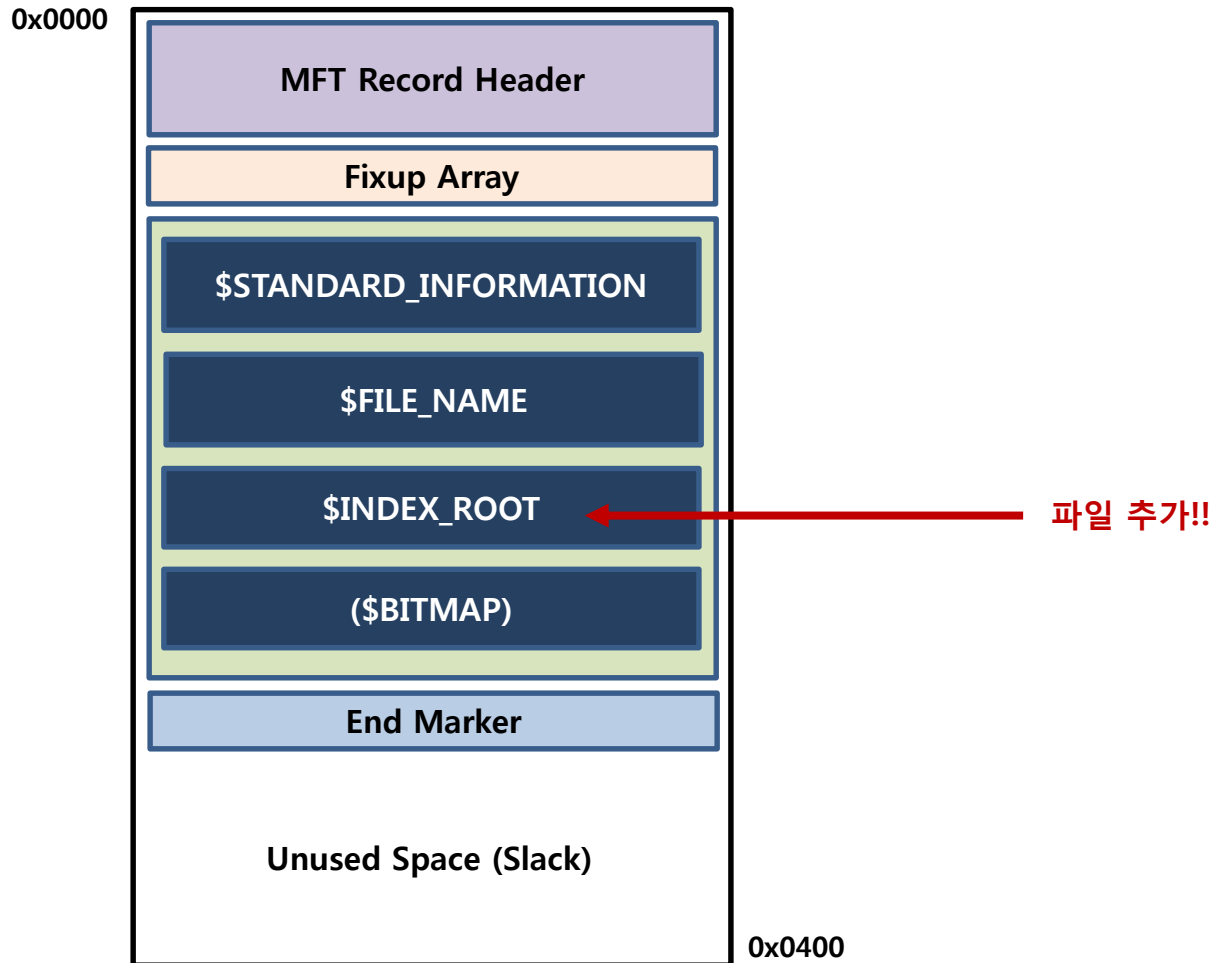
- MFT 레코드 구조 → **폴더**



은닉 데이터 분석

슬랙 공간 분석

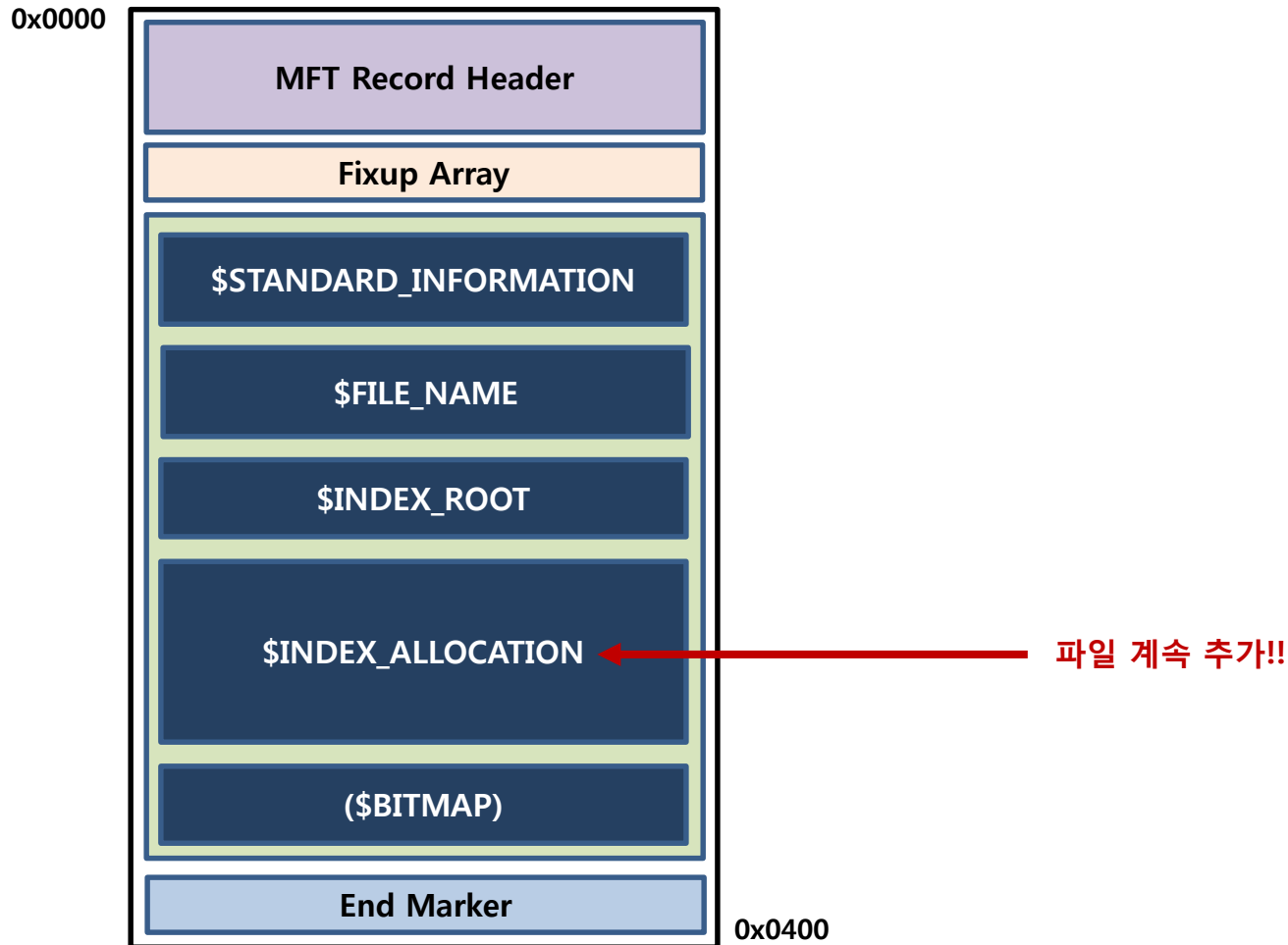
- MFT 레코드 구조 → **폴더**



은닉 데이터 분석

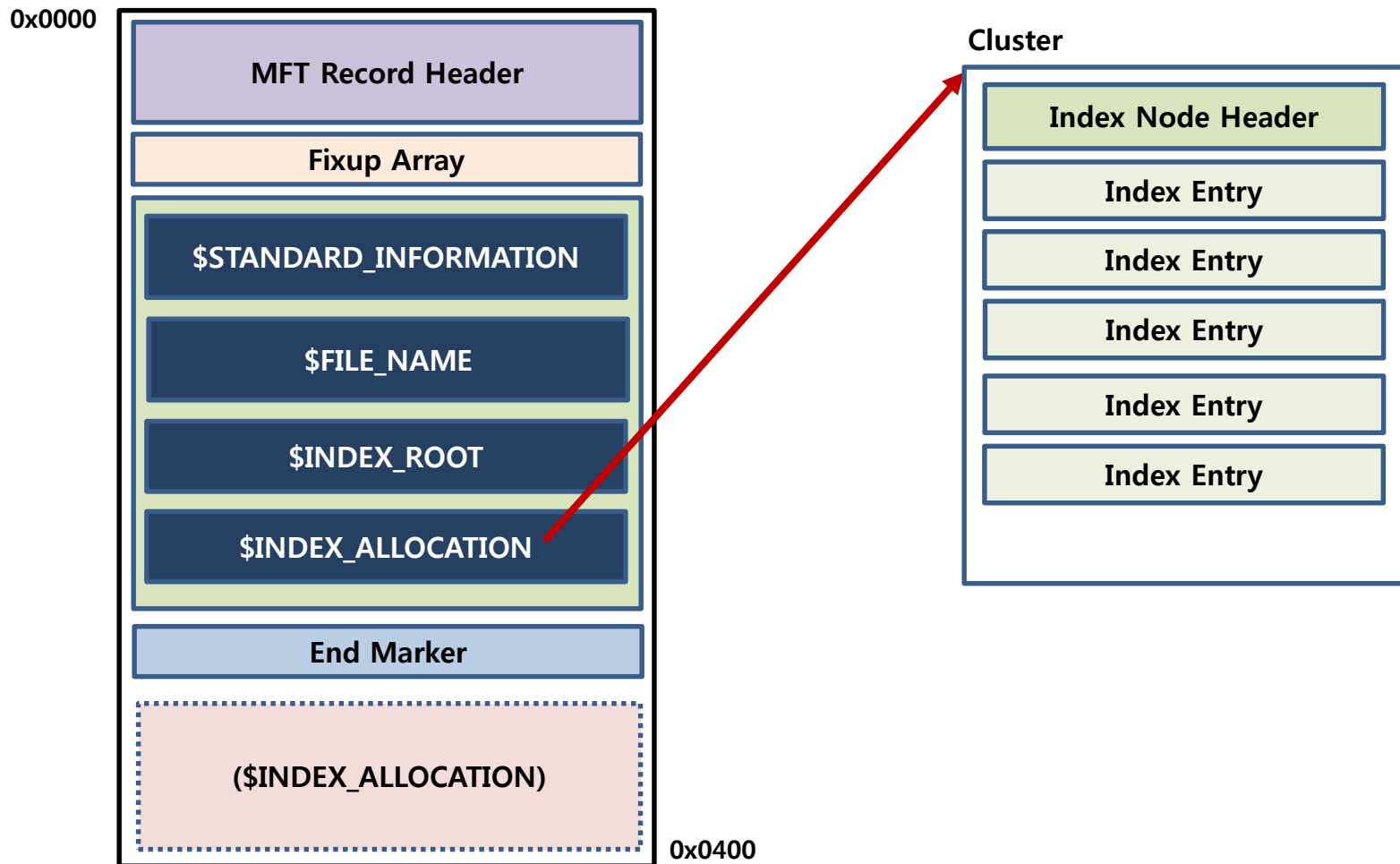
슬랙 공간 분석

- MFT 레코드 구조 → **폴더**



슬랙 공간 분석

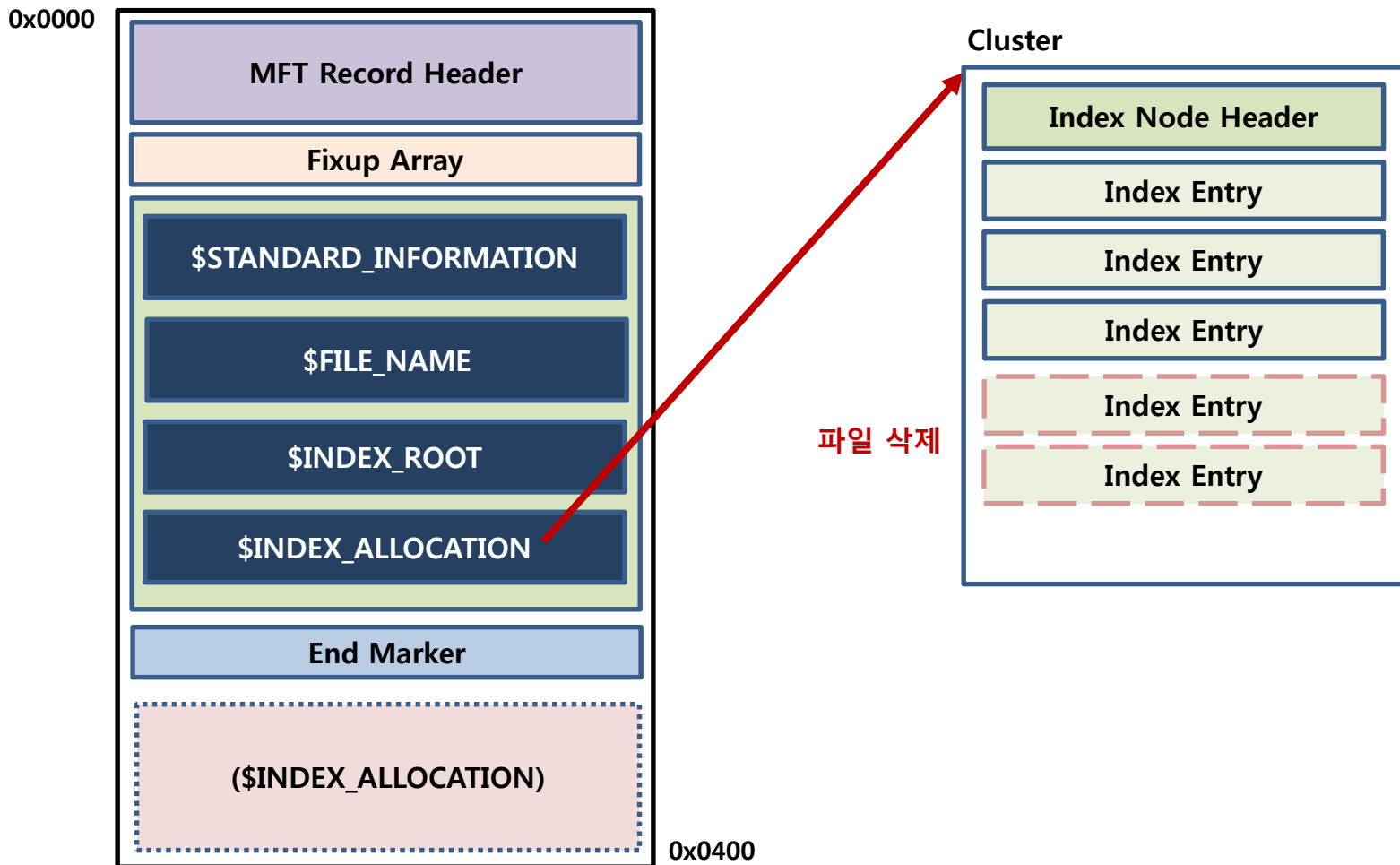
- MFT 레코드 구조 ➔ **폴더**



은닉 데이터 분석

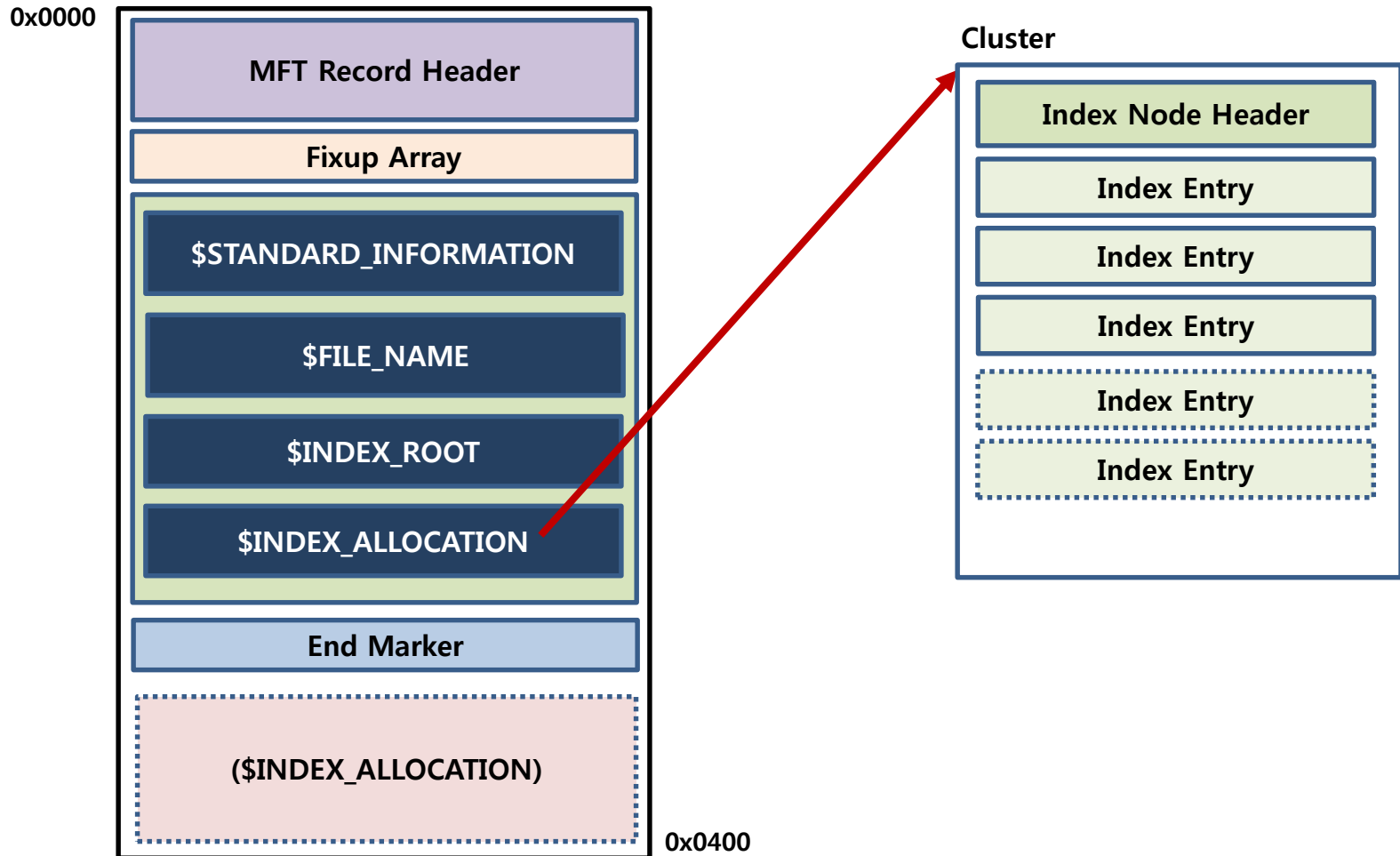
슬랙 공간 분석

- MFT 레코드 구조 ➔ 폴더



슬랙 공간 분석

- MFT 레코드 구조 ➔ **폴더**



슬랙 공간 분석

- **INDX 슬랙 분석 도구**

- **Windows INDX Slack Parser (wisp) – TZWorks**

- ✓ https://www.tzworks.net/prototype_page.php?proto_id=21

- **INDX 슬랙 조사에 활용하기!!!**

1. \$Recycle.Bin 내에 파일 생성
2. 파일 삭제 후 INDX 정보 확인

은닉 데이터 분석

슬랙 공간 분석

- MFT 레코드 구조 ➔ 폴더

메타데이터 조작

▪ \$Boot

- \$Boot는 부트 섹터의 내용을 저장 (VBR의 부트 섹터 위치를 가르킴)
- \$Boot 파일 크기는 제한이 없음 → 크기를 늘려 데이터 은닉

▪ \$BadClus

- \$BadClus는 배드섹터가 포함된 클러스터를 관리
- 정상 클러스터를 \$BadClus에 등록한 후 의도적인 데이터 저장

대체 데이터 스트림

Record Header	Fixup Array	\$STD_INFO	\$FNA	\$DATA Main Stream	\$DATA Alternative Stream	End Marker	Unused Space
---------------	-------------	------------	-------	--------------------	---------------------------	------------	--------------

- MAC 클라이언트를 지원하기 위한 기능으로 \$DATA 속성을 2개 이상 가질 수 있음
- ADS 속성을 고유한 이름을 통해 접근 ➔ \$DATA 속성의 이름을 가져야 함
- ADS를 데이터 은닉에 활용 ➔ 악성코드에서 활용한 예

- **ADS 활용**

- ✓ W005DocumentSummaryInformation
- ✓ W005SummaryInformation
- ✓ Zone.Identifier
- ✓ Thumbs.db.encryptable

대체 데이터 스트림

- 파일에 ADS 생성

```
$> echo "This is ADS" > proneer.txt:ads.txt  
$> type c:\windows\notepad.exe > proneer.txt:ads.exe
```

- 폴더에 ADS 생성

```
$> echo "This is attached to directory list" > :ads3  
$> echo "This is malware" > C:\Windows\System32\svchost.exe
```

- ADS 내용 확인

```
$> more < proneer.txt:ads1
```

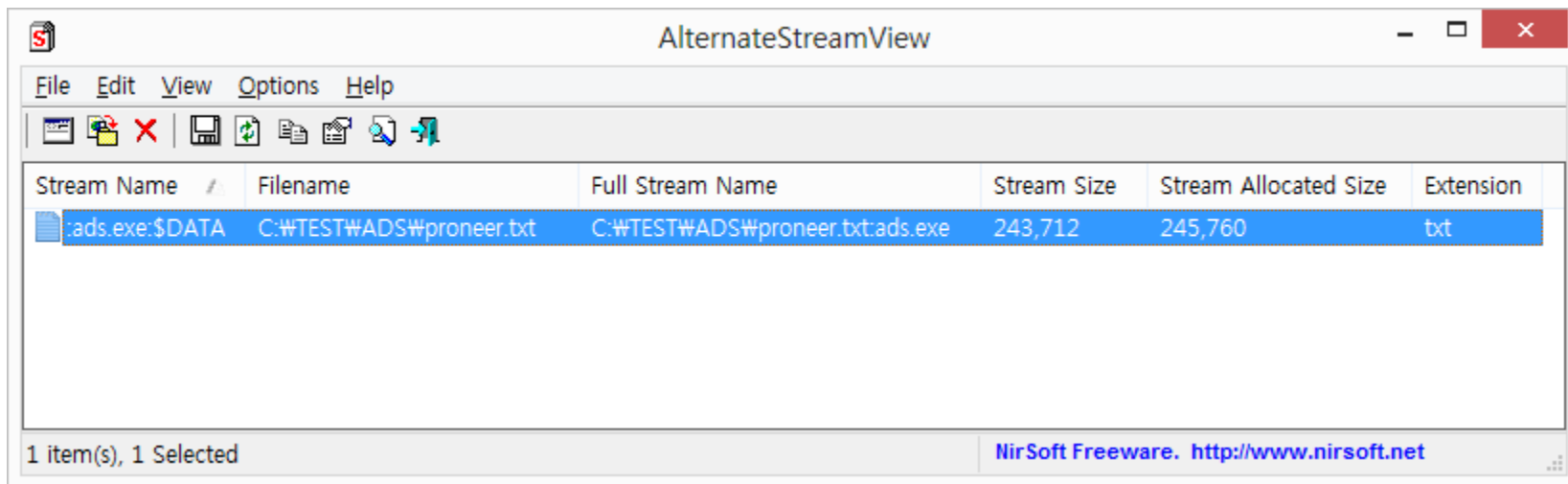
- ADS 존재 여부 확인

```
$> dir /R [folder]
```

대체 데이터 스트림

■ ADS 삭제

- NTFS가 아닌 볼륨에 복사
- 메인스트림 삭제
- ADS 관련 도구 이용 (AlternateStreamView, http://www.nirsoft.net/utils/alternate_data_streams.html)



로그 분석

파일시스템 로그

- NTFS 파일시스템 로그

- %SystemDrive%\\$LogFile
- %SystemDrive%\\$Extend\%\$UsnJrnl:\$J

- 파일시스템 로그의 장점

- 특정 기간 동안 일어난 파일시스템 이벤트 분석 가능
- 삭제된 파일의 흔적 추적

\$LogFile

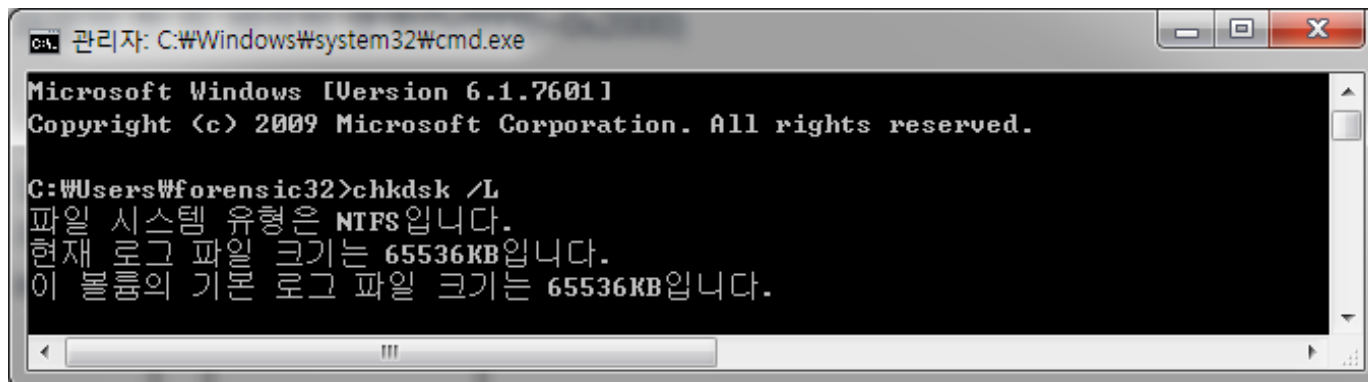
- **트랜잭션 로그 파일**
 - 비정상적인 시스템 오류에 대응하기 위한 로그
- **트랜잭션 단위의 로그 기록**
 - 파일/디렉터리 생성
 - 파일/디렉터리 삭제
 - 파일/디렉터리 변경
 - MFT 레코드 변경

\$LogFile

▪ 기록되는 로그의 양

- 일반적으로 64MB 크기 → PC의 일반적 작업이라면 2~3시간 정도의 로그가 보관
- 침해사고 준비도 측면에서 용량 증가 필요
- 크기 조절

```
$> chkdsk /L:[size]
```



```
관리자: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

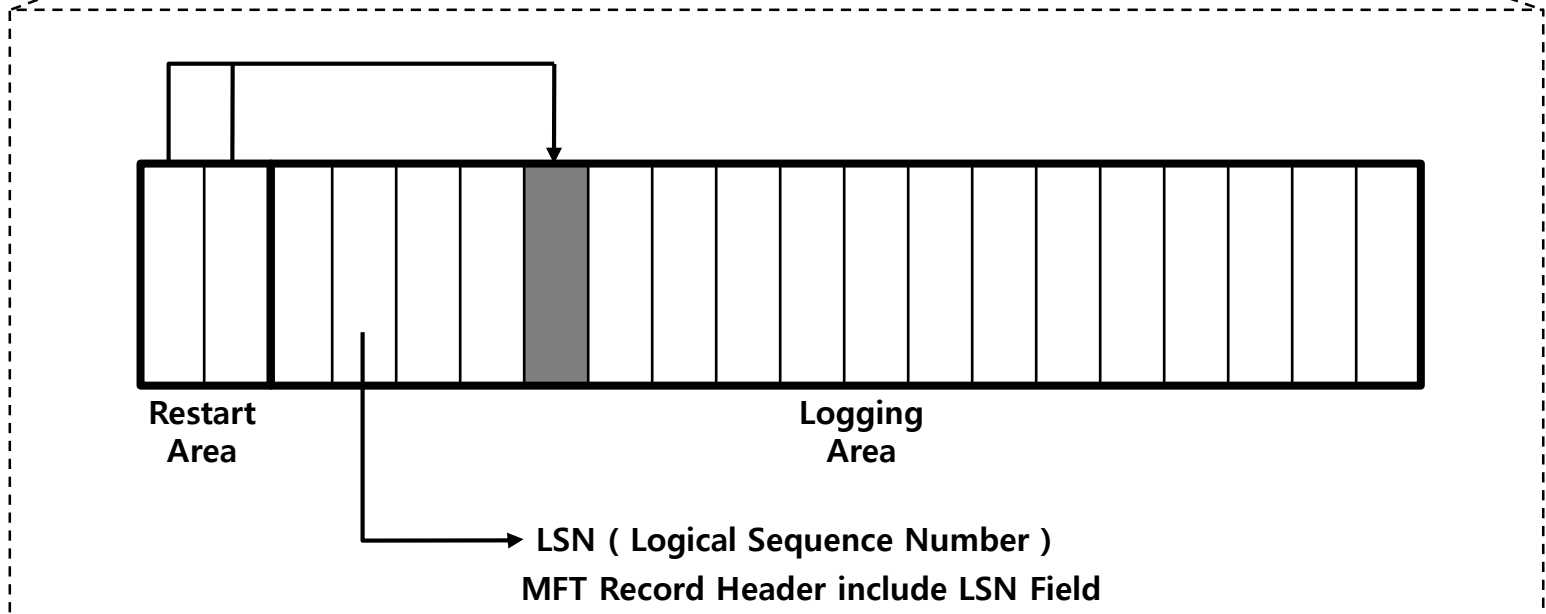
C:\Users\forensic32>chkdsk /L
파일 시스템 유형은 NTFS입니다.
현재 로그 파일 크기는 65536KB입니다.
이 볼륨의 기본 로그 파일 크기는 65536KB입니다.
```


\$LogFile

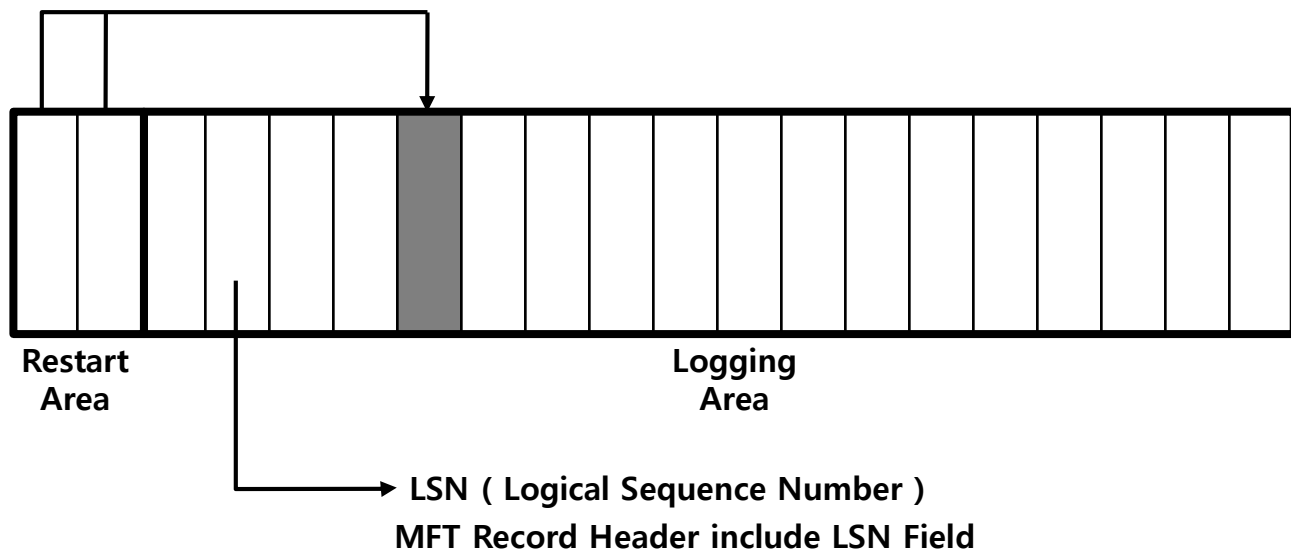
\$LogFile
MFT Record
(#2)



Non-resident \$DATA Attribute



\$LogFile



■ 재시작 영역

- 파일의 첫 두 페이지 영역 (0x0000~0x2000)
- 마지막 작업 레코드 지정

■ 로깅 영역

- 작업 레코드 기록
- 버퍼 페이지 영역과 일반 페이지 영역으로 구분

\$LogFile

- **\$LogFile 트랜잭션 레코드 분석**
 - 각 레코드는 파일 단위의 레코드가 아님
 - 생성, 삭제, 변경 등의 이벤트 단위로 레코드 결합
 - **분석 대상 이벤트**
 - ✓ 파일 생성
 - ✓ 파일 삭제
 - ✓ 파일 데이터 변경
 - ✓ 파일명 변경
 - ✓ 파일 이동

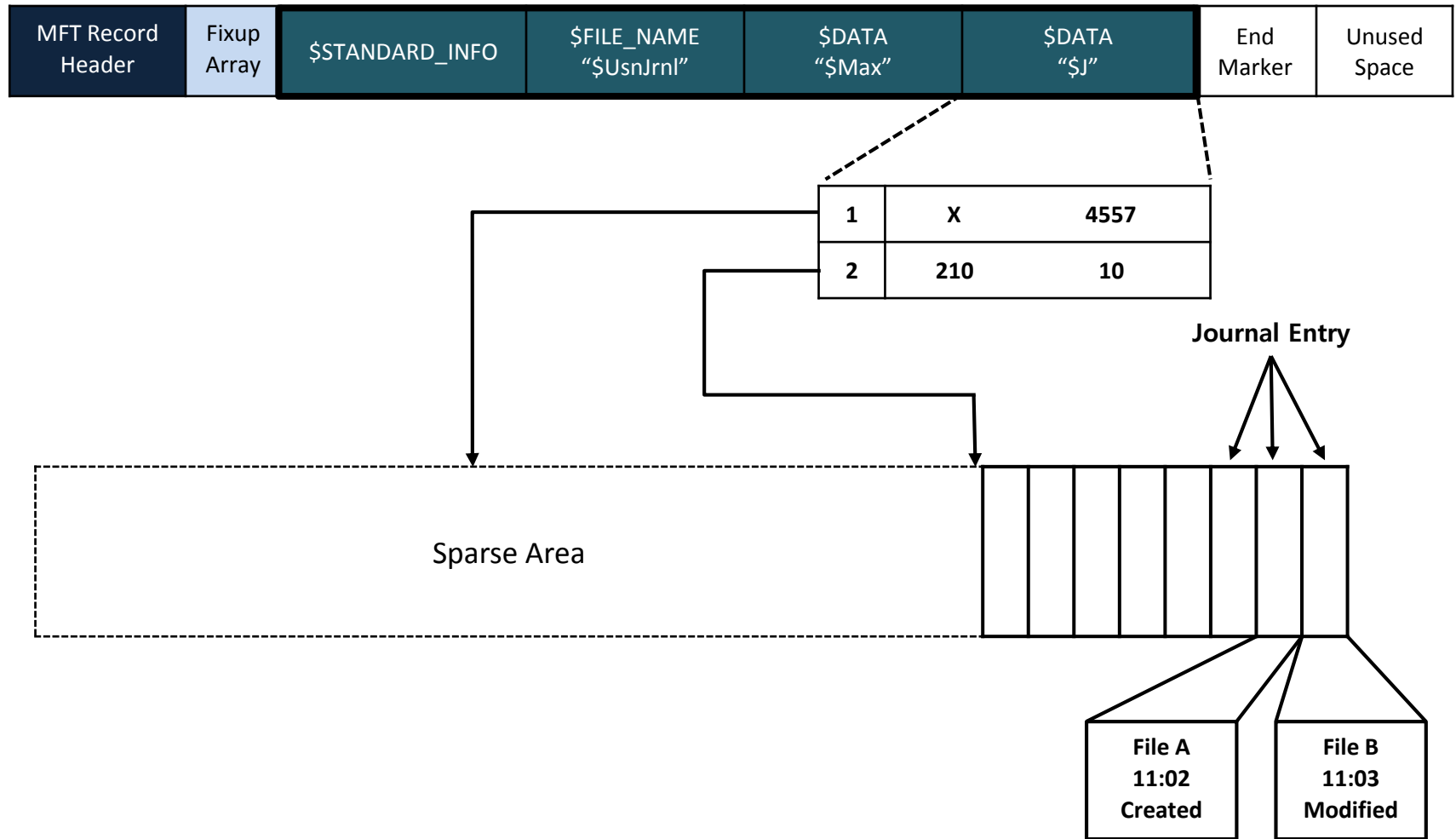
\$UsnJrnl

- NTFS 변경 로그
 - 파일이나 디렉터리의 변경 내용 기록
 - 윈도우 7부터 기본 활성화
- 기록되는 로그의 양
 - 컴퓨터를 계속 사용할 경우, 1~2일의 로그 저장
 - 하루 8시간 정도만 사용할 경우, 4~5일의 로그 저장

\$UsnJrnl

- \$Max와 \$J
 - \$UsnJrnl 파일은 2개의 ADS를 가짐
 - \$Max
 - ✓ 변경 로그에 대한 메타데이터
 - \$J
 - ✓ 실제 변경 레코드

\$UsnJrnl



실습 #2

- NTFS 파일시스템 로그 통합 분석하기!!!
 - \$MFT
 - \$LogFile
 - \$UsnJrnl

파일시스템 터널링

터널링

- 삭제된 파일의 생성 시간 캐시
- 짧은 시간(기본 15초) 내에 동일 이름의 파일을 삭제하고 다시 생성하면 생성 시간 유지
- **터널링 사용 이유**
 - 파일 내용 변경 시 임시파일을 사용하는 경우, 파일 동일성 유지
- **터널링 설정**
 - 캐시 시간
 - ✓ HKLM\SYSTEM\CurrentControlSet\Control\FileSystem\MaximumTunnelEntryAgeInSeconds
 - 터널링 사용 여부
 - ✓ HKLM\SYSTEM\CurrentControlSet\Control\FileSystem\MaximumTunnelEntries
 - 0 : 비활성화
 - 1 : 활성화

파일시스템 터널링

터널링

```
C:\WINDOWS\system32\cmd.exe

C:\Tunneling>echo > file1

C:\Tunneling>dir /tc
Volume in drive C has no label.
Volume Serial Number is 941C-9471

Directory of C:\Tunneling

02/08/2013  08:41 PM    <DIR>          .
02/08/2013  08:41 PM    <DIR>          ..
02/08/2013  08:42 PM                13 file1
               1 File(s)                13 bytes
               2 Dir(s)  6,156,234,752 bytes free

C:\Tunneling>echo %time%
20:44:16.07

C:\Tunneling>ren file1 file2

C:\Tunneling>dir /tc
Volume in drive C has no label.
Volume Serial Number is 941C-9471

Directory of C:\Tunneling

02/08/2013  08:41 PM    <DIR>          .
02/08/2013  08:41 PM    <DIR>          ..
02/08/2013  08:42 PM                13 file2
               1 File(s)                13 bytes
               2 Dir(s)  6,156,218,368 bytes free

C:\Tunneling>echo > file1

C:\Tunneling>dir /tc
Volume in drive C has no label.
Volume Serial Number is 941C-9471

Directory of C:\Tunneling

02/08/2013  08:41 PM    <DIR>          .
02/08/2013  08:41 PM    <DIR>          ..
02/08/2013  08:42 PM                13 file1
02/08/2013  08:42 PM                13 file2
               2 File(s)                26 bytes
               2 Dir(s)  6,156,218,368 bytes free

C:\Tunneling>echo %time%
20:44:33.57

C:\Tunneling>_
```

실습 #3

- 파일시스템 터널링 확인하기!!!

파일시스템 갱신 지연 시간

파일시스템 갱신 지연 시간

NTFS 1시간, FAT 1일 법칙

- 파일시스템 별 갱신 지연 시간

- FAT 파일시스템

- ✓ 생성 시간 : 10 밀리초(ms)

- ✓ 수정 시간 : 2 초(sec)

- ✓ 접근 시간 : 1 일(day)

- NTFS

- ✓ 접근 시간 : 1 시간(hour)

- 갱신 지연 시간 이내에 행한 파일의 생성, 수정, 접근은 파일시스템에 반영되지 않음

챌린지

#1 – CODEGATE 2011 – F200

#2 – DC3 2013, 303 - Alternate Data Stream (v1.1b)

- 303ADS_ChallengeFiles.001

1. What method was used to hide the files?
2. What are the messages in the hidden files?
3. What are the names of the host and embedded files?
4. What is the command to create an ADS?

