기술문서

Encase Basic

행정4 김범연

ccibomb@gmail.com

ccibomb.tistory.com

2010. 5. 8.

○ 목 차 ○

0. Encase 개요[i]

Encase 는 Guidence 사에서 만든 컴퓨터 포렌식스 툴이다. 현재 세계 점유율이 1 위이며, 사실상의 표준이기도 하다.(미국 FBI 의 지원으로 인해서, 회사가 커졌으며, 현재 미국법원에서 Encase 로 제출하는 자료에 대해서 증거로 인정되면서, 그 세력이 더 커졌다고 볼 수 있다.)

하지만, 실제로 Encase 가 언제나 최고의 툴은 아니다. FTK(Forensic Tookit , accessData) 의 경우, 일본에서는 Encase 보다 많이 사용되고 있다. 그러나 세계적으로 본다면, 미국 법률시장이 가장 크므로, Encase 는 가장 보편화된 툴이라고 할 수 있다. 그리고 현재는 하나의 툴이 아니라, 두 개 이상의 툴로, 결과를 검증 할 수 있어야 하기 때문에, Encase 든 FTK 든 제대로 배워두어야 제대로 된 수사를 할 수 있다.

1) Encase 제품군

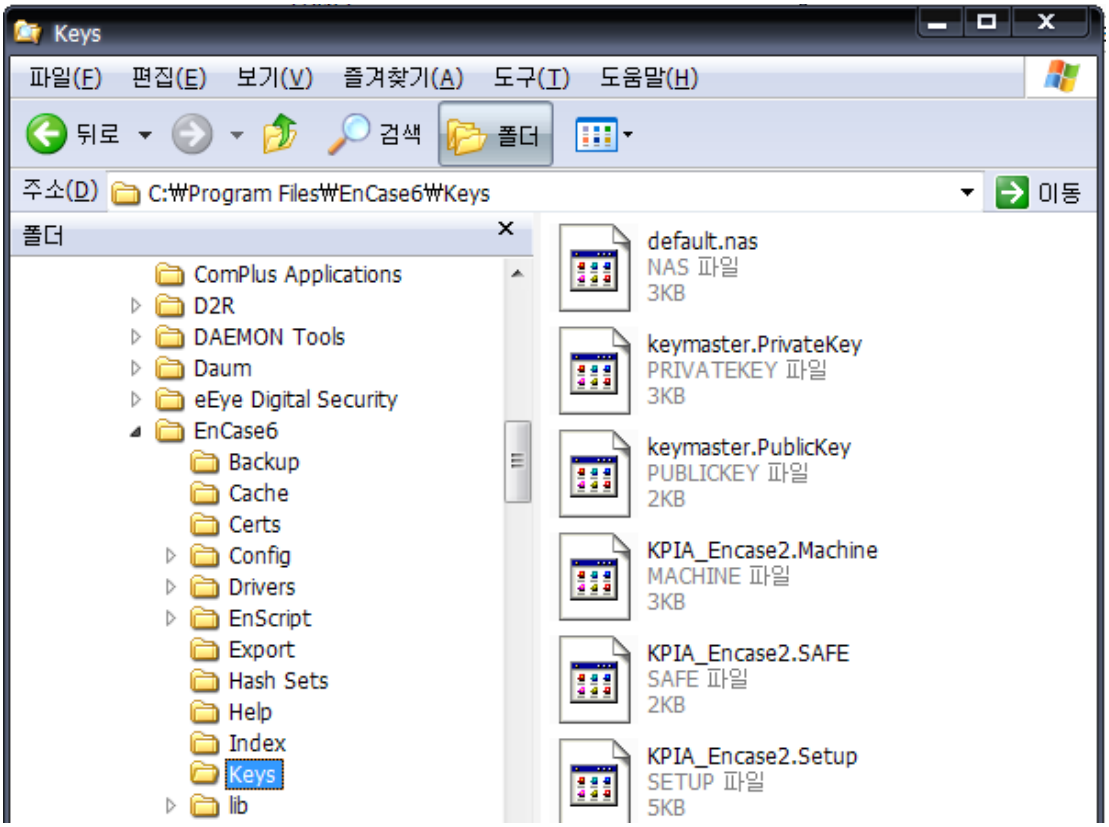
- Encase Forensic : 로컬 컴퓨터 하드디스크의 기본 분석 도구
- Field Intelliegence Model(FIM) : 네트워크 컴퓨터 및 서버의 휘발성 자료를 포함한 증거이미지 획득 및 조사가 가능, 수사 기관용으로 Enterprise 축소판
- Encase Enterprise : 기업의 Live 시스템 및 파일에의 접근과 분석이 가능하여 사고대응 시간을 줄일 수 있음. 관제로 활용

2) Encase 지원 파일 시스템

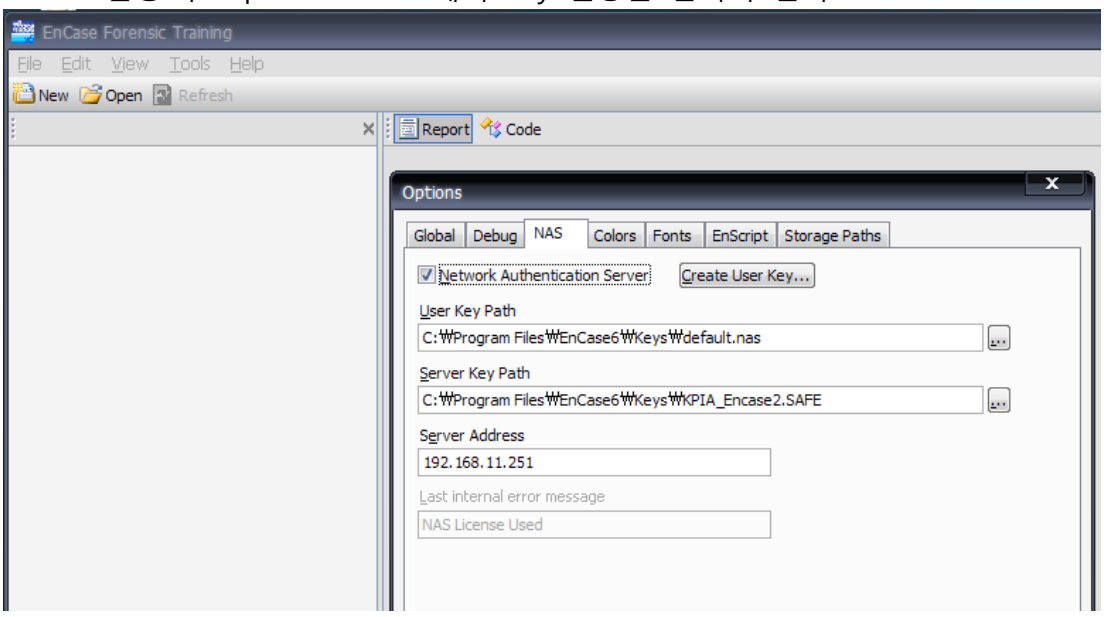
- FAT12, FAT16, FAT32
- NTFS
- EXT2, EXT3
- CDFS
- HFS, HFS+ (MAC 파일 시스템)
- PALM (Palm-PAD 파일 시스템)
- UFS (Unix 파일 시스템)

1. Encase 설치

1. Encase 가 설치된 폴더의 Keys 폴더에 User Key 와 Server Key 를 넣은 후,



2. Encase 실행 후 Options - NAS 에서 Key 인증을 받아야 한다.



NAS 탭에서는 서버로부터 Encase 동글키를 부여받는 network Authentication 을 위한 모든 설정을 할 수 있다.

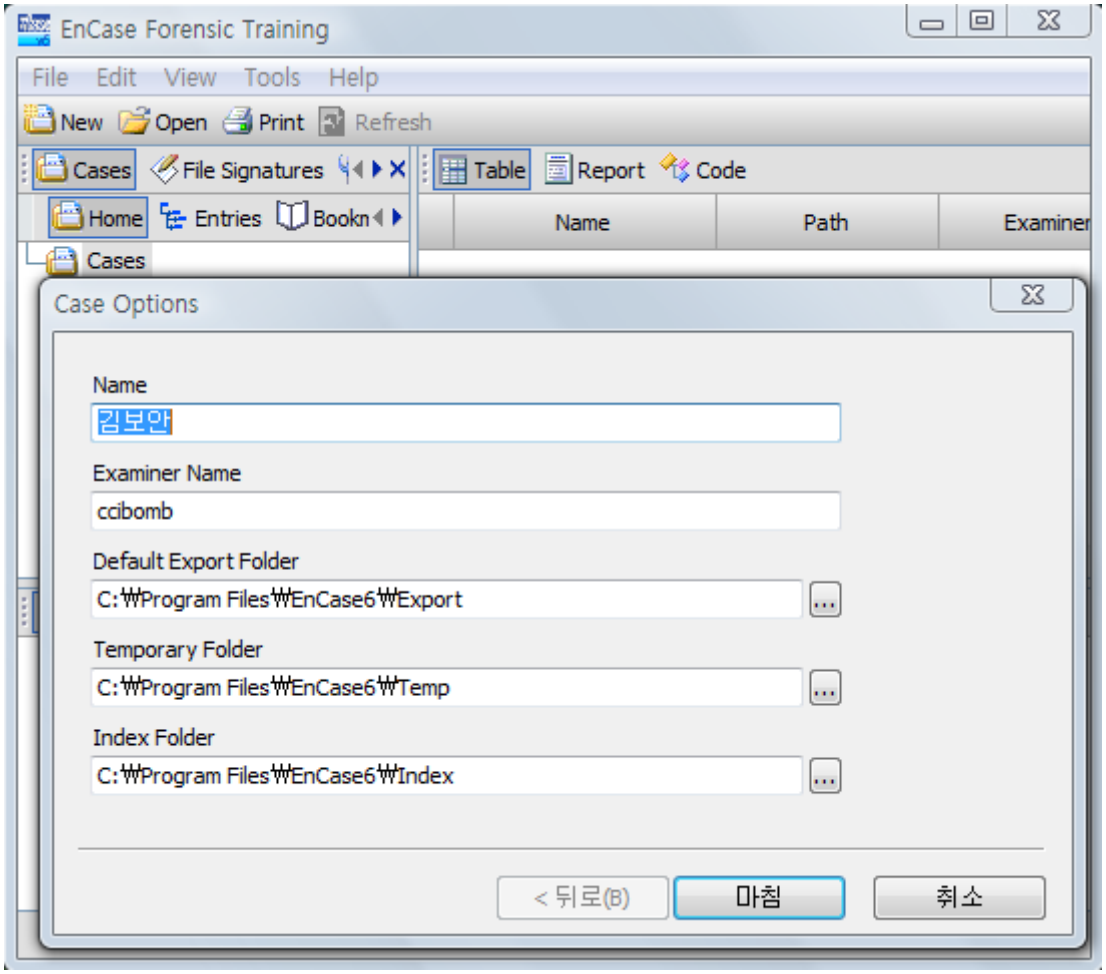
2. Case 관리

Encase 의 가장 강력한 기능 중 하나는 다른 타입의 매체들을 함께 구성할 수 있다는 점이다. 이를 통해, 개개의 독립적인 검색이 아니라 각각을 하나의 단위로서 검색이 가능하다. 이러한 과정은 시간을 절약하게 해주며, 조사관이 증거의 조사에 집중할 수 있도록 해준다.

Encase 에서 Case 라는 것은 어떤 사건 하나를 관리하는 단위이다. 실제로 작업을 저장하면 ".Case" 파일이 생기게 되는데 현재 상황, 검색된 결과들, 북마크 결과들, 파일 스캔 결과들이 저장되게 된다.

조사를 시작하고 매체를 획득하기 전에, Case 생성 후 어떻게 접근할 것인지 고려해야 한다. 동시에 수 명의 조사관이 해당 정보를 보아야 할 필요가 있을 수 있다. 이러한 경우, 증거이미지(evidence files)은 중앙 파일 서버에 위치시키고, Case 사본 파일들을 각 조사관들의 컴퓨터에 위치시켜야 한다(Case 파일은 한 번에 한 명 이상의 접근을 거부하기 때문이다).

Encase 에서의 모든 작업은 일단 Case 를 생성한 다음에 실행하게 된다.



위의 그림은 **File->New** 를 클릭하거나, 툴바의 **New** 아이콘을 클릭하여 나타난 Case Option 이다. 먼저 Name 을 지정하는 것이 보이는데, 이것은 이 Case 의 이름을 지정하는 것이다. 예를 들면, "김보안", 이런식으로 말이다. 그 다음은 분석관의 이름이다. "ccibomb", 이 두 개는 간단하다. 그리고 이제 나머지 세 개의 폴더에 대해서 궁금증을 가져야 한다. Default Export Folder, Temporary Folder, Index Folder 각각 무엇을 말하는 것일까?

- Default Export Folder

Encase 에서 파일을 복구할때(삭제되었거나 이미지 안에서 실제 파일로 저장할 때) 이를 기본적으로 어디에 생성할 것인가를 지정하게 된다.

- Temporary Folder

Encase 에서 압축 파일을 보여준다거나, 어떤 내용을 임시로 보여주기 위해서 Temp 폴더로 사용하는 공간이다. 윈도우의 Temp 라고 생각하면 된다. 조사과정에서 생성되는 임시 파일들의 분류 및 관리를 하기 위해 필요하다.

- Index Folder

Encase V6 부터는 문서 타입의 파일에 대해서 Index Search 를 지원한다. 예를 들어 txt 나, 오피스 종류의 문서의 텍스트를 추출해서 이것에 대한 단어들에 인덱스를 만들고 나중에 키워드로 검색할 때, 이 인덱스에서 찾아주는 것이다.

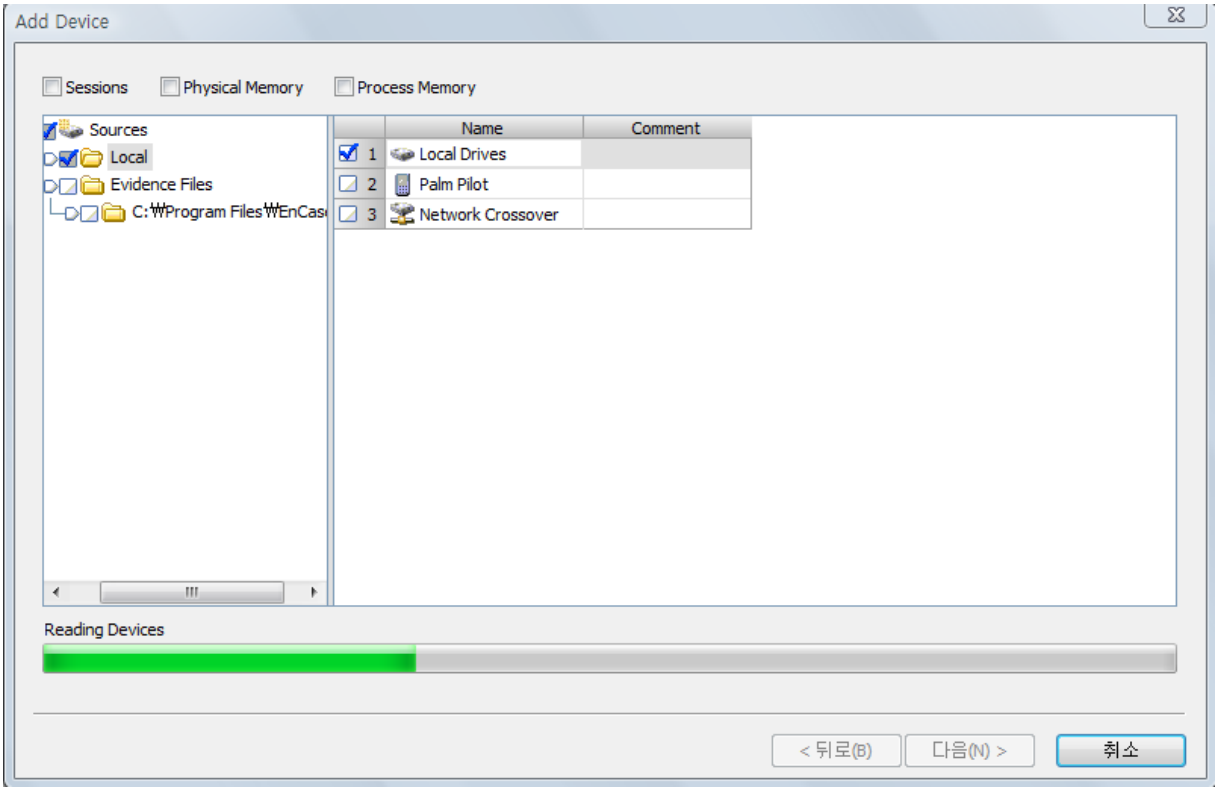
위의 폴더 들이 구분되어 있는데, 사실 이 폴더들의 의미보다 더 중요한 것은, 바로 왜 이렇게 폴더들이 나뉘어져 있는가, 그리고 왜 공통적으로 지정할 수 도 있는데(기본 설정은 되어있다.) 왜 Case 생성시에 설정하게 되어있는가 라고 생각한다. 포렌식이라는 것은 원본의 훼손을 막아야 하므로, 혹시나 윈도우 임시폴더에 어떤 데이터를 저장하게 됨으로써 사용자 PC 의 데이터 훼손이 되지 않도록 고려하는 것이다. 즉, 사용자가 지정한 부분에만 데이터를 쓰게 하는 것이다. 그래서 Encase 의 경우 Registry 도 전혀 쓰지 않는다.

Encase Forensic 방법론에서는 조사관이 2nd Hard drive 를 사용하거나, 적어도 디지털 증거로 사용하려는 부트 Hard drive 와는 별도의 2nd Partition 을 이용할 것을 추천한다. 또한, 개개의 폴더나 범죄와 관련되는 것으로 보이는 데이터만이 아니라 Hard drive 전체 또는 Partition 전체를 이미지 뜨기를 추천하는데, 이는 사건 조사시 Cross-contamination 을 방지하고 조사 이후 필요한 Data 가 부재를 방지하기 위함이다.

3. Add Device[ii]

케이스를 생성했다면, 실제로 증거물을 분석해 보아야 한다.

Encase 에서 증거물의 추가는 굉장히 쉽다. 그냥 툴 바에 있는 Add Device 를 선택하면 증거물의 추가가 가능하다. (케이스를 생성해야만 Add Device 가 가능하다.)



여기서 일단 Local Drives 를 선택한다. 팜 파일럿이나 네트워크 연결도 가능하나, 매체의 종류와 상관없이 증거 이미지(evidence file)의 생성 과정은 동일하므로 실제 현재 연결되어 있는 디스크를 추가해볼 것이다. 밑을 보면 증거 이미지를 추가할 수 있는 디렉토리도 보인다.

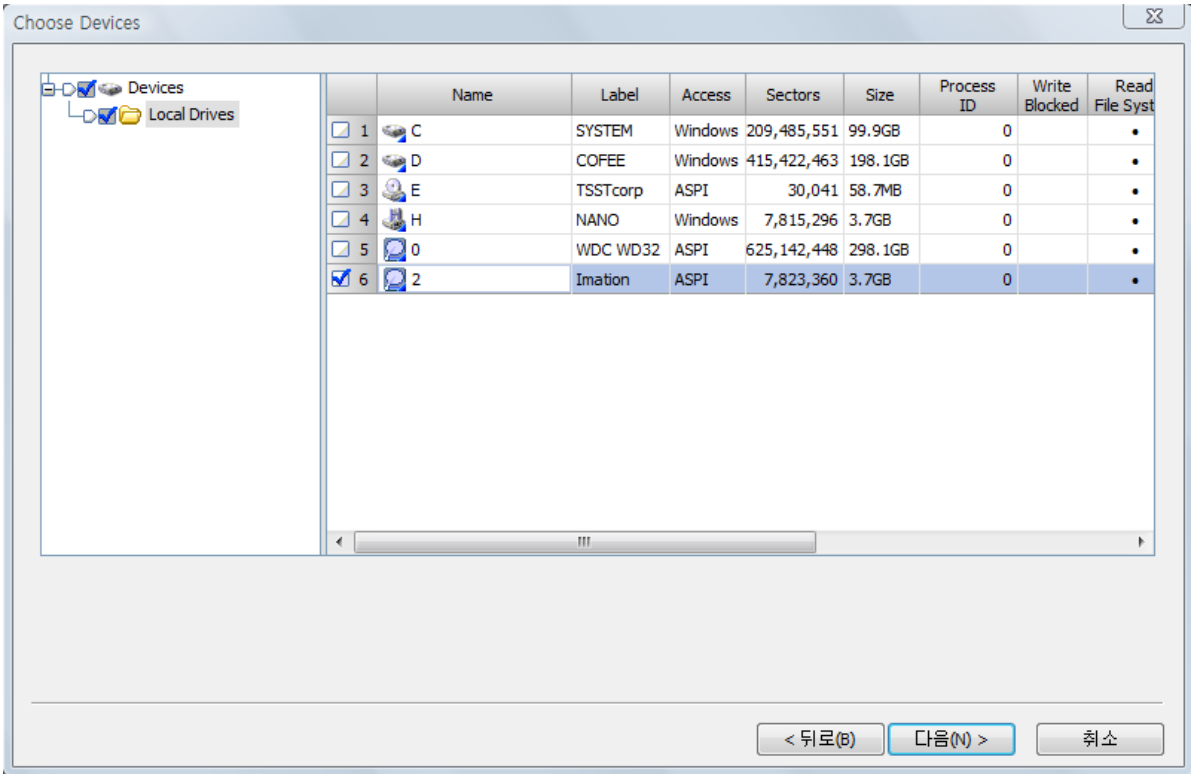
Locals

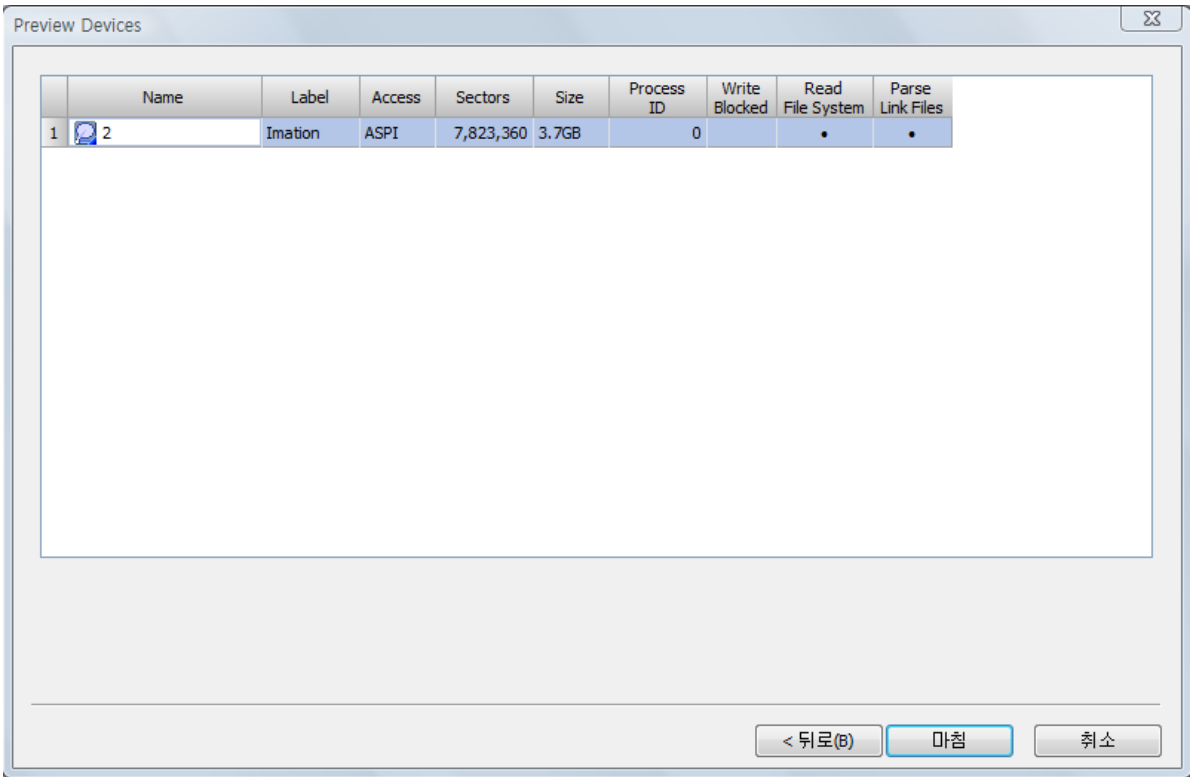
- 실제로 하드웨어에 물려있는 장치들

Evidence Files

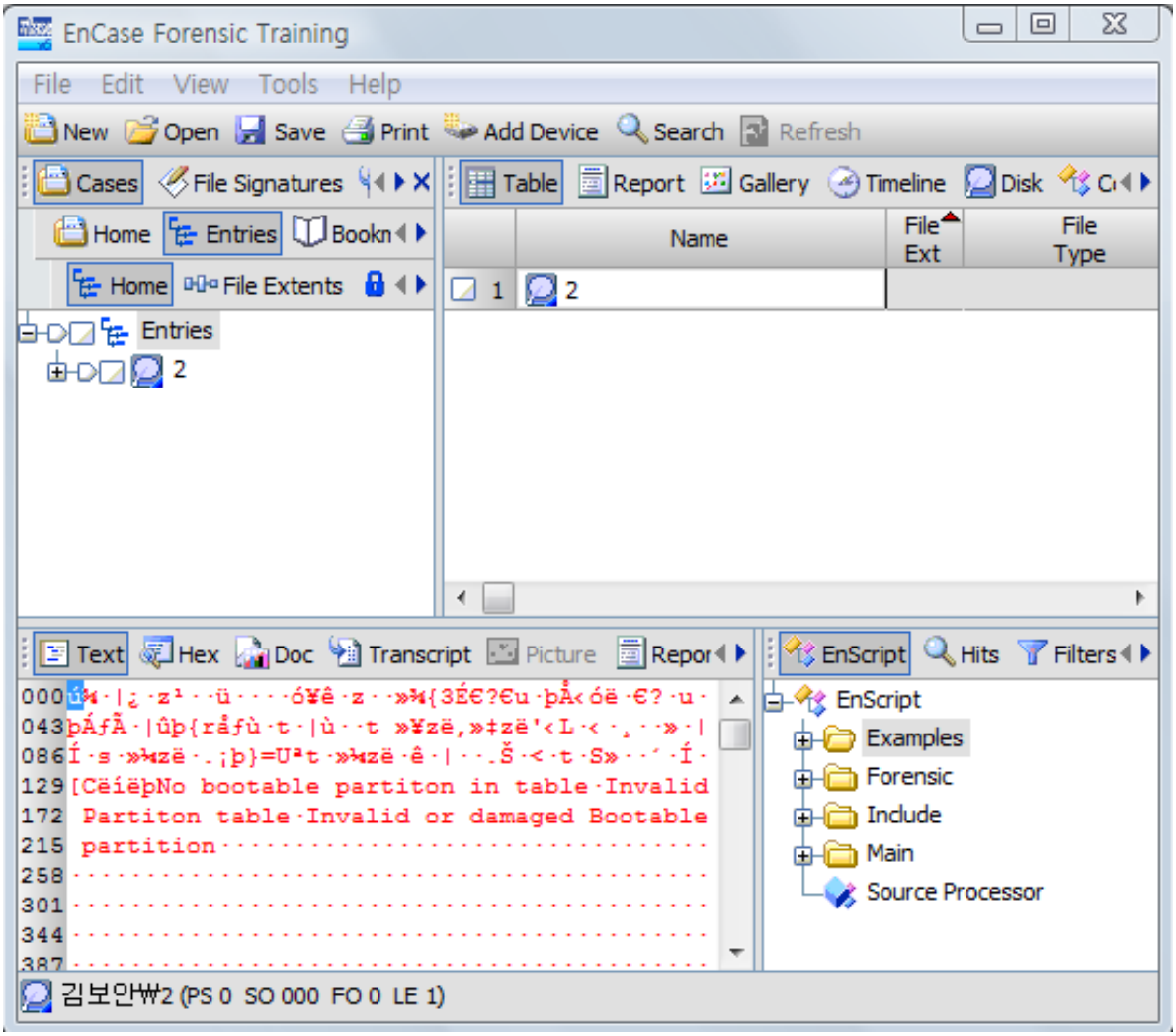
- Locals 을 증거이미지로 만든것

그러면 이제 실제로 디스크를 추가할 수 있는 창이 뜨게 된다. 이중에서 알파벳으로 된것은 Logical Drive 들이고 0, 1, 2 숫자는 Physical Drive 를 말한다.



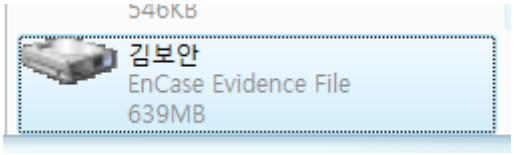


해당 디스크에 대한 정보가 나오게 된다. 마침을 누르면 실제로 파일 시스템을 분석하기 시작한다. 이게 모두 끝나면 이제 실제 파일 시스템의 내용을 볼 수 있다.

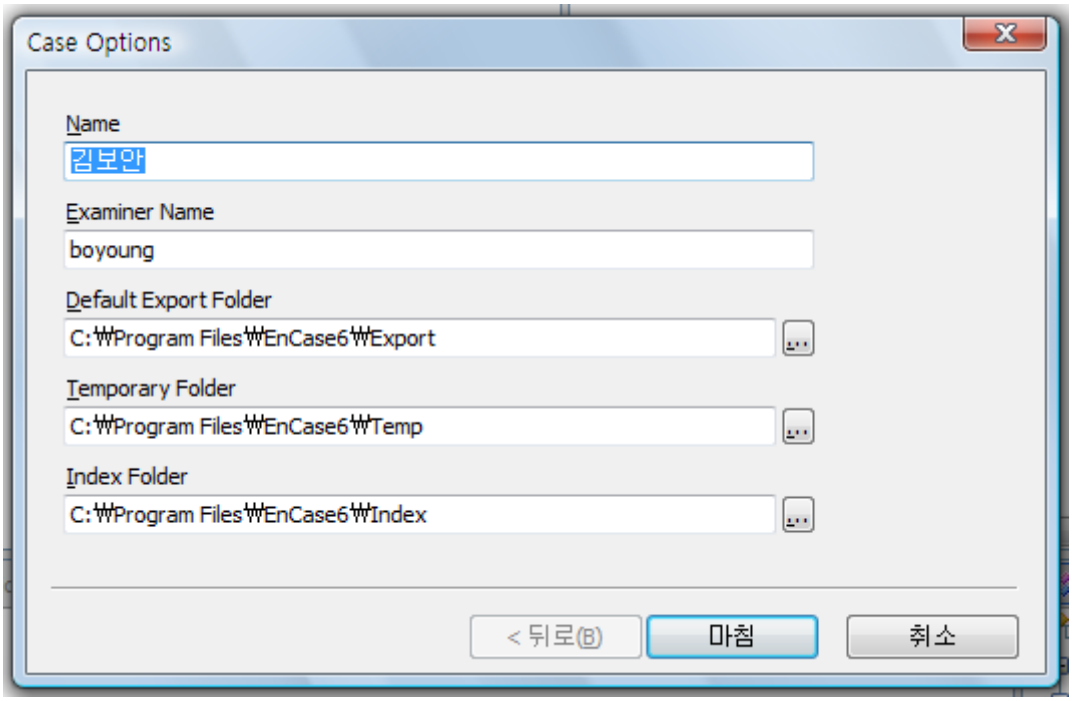


4. 기존 Case 파일 분석

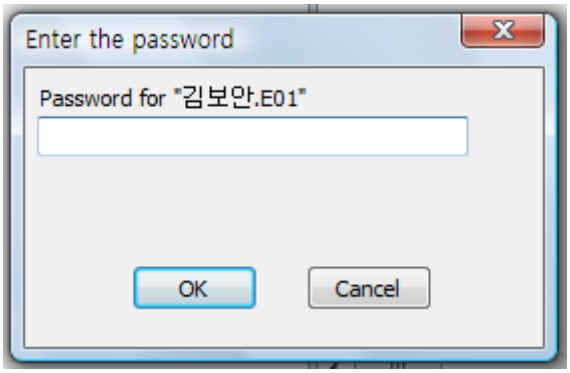
1. 증거이미지를 Encase 의 tree pane 에 끌어다 놓는다.



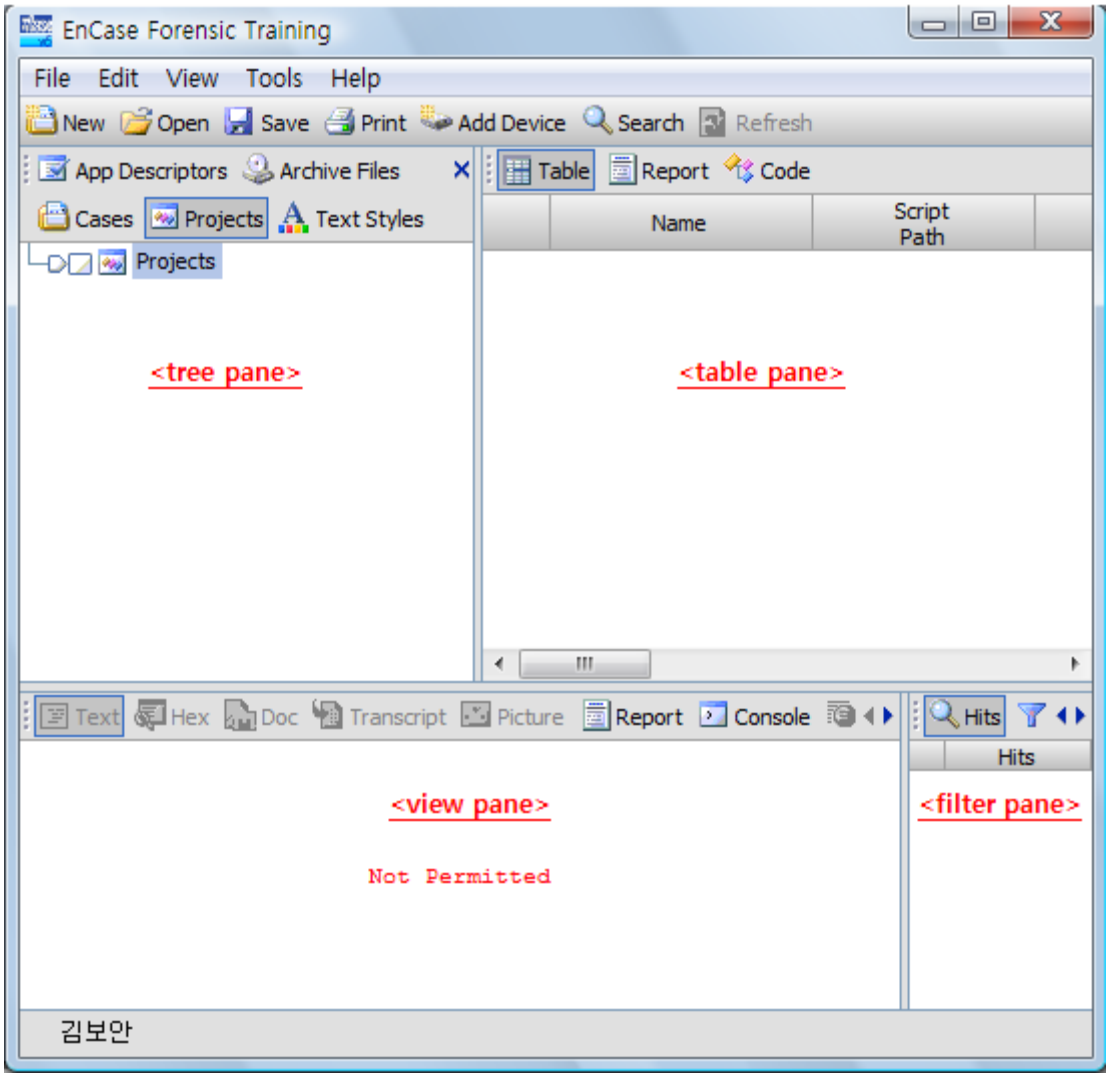
2. New case 로 등록한다.



3. Password 가 걸려있는 경우, 해당 Password 입력



4. 화면 구성 : tree pane, table pane, filter pane, view pane (왼쪽 위부터 시계방향)



- tree pane(좌상) - 파일들을 tree 모양으로 보여줌
- Table pane(우상) - 선택한 파일들의 속성, 등등의 정보
- view pane(좌하) - 데이터를 자세히 보는 데 쓰임
- filter pane(우하) - 자동화 작업, Keyword Searching, 스크립트

5. Encase Concepts

5-0. 증거 이미지 (Evidence File)

Encase 방법론의 가장 중요한 요소는 바로 증거 이미지 (Evidence file)이다. 이 파일은 세 가지 기본 요소(헤더, 체크섬, 데이터 블록)로 구성되는데, 이들은 분석시 컴퓨터 디스크의 상태를 Self-checking하여 제공한다.

5-1. CRC (Cyclical Redundancy Check)

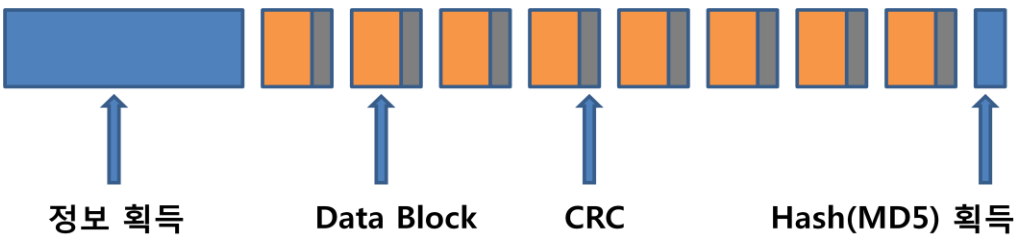
CRC는 체크섬과 매우 비슷한 방식으로 동작하는데, 체크섬과 달리 Order-sensitive 특성을 지닌다. 즉, 문자열 '1234'와 '4321'은 같은 체크섬을 갖지만, 서로 다른 CRC를 가진다. 대부분의 Hard drive는 각 섹터마다 하나의 CRC를 저장한다. 디스크에서 Read 에러가 발생하였다는 것은 디스크 섹터의 CRC와 해당 섹터가 읽어진 후 drive hardware에 의해 재계산된 값이 서로 일치하지 않음을 의미한다.

5-2. 증거이미지 포맷 (Evidence File Format)

각각의 파일은 정확하게 섹터 대 섹터 사본이다. 파일이 생성될 때, 사용자는 조사와 관련된 정보를 제공하게 된다. Encase는 이러한 정보 및 증거이미지 안에 있는 다른 정보들을 아카이브에 넣는다. 각 파일은 Data block마다 32-bit CRC를 이용하여 인증받음으로써, 조사관은 법정에 해당 증거를 제출할 수 있다.

Encase는 디스크 이미지 전체에 대한 CRC 값이 아닌, 개개의 블록마다 CRC를 계산하여 그 무결성과 속도를 높인다. 어느 파일이나 어느 섹터에서 에러가 발생하였는지 그 위치 확인도 가능하다.

또한, Encase는 Physical drive나 Logical drive 의 이미지 생성시, MD5 해쉬값을 계산하여 증거이미지에 그 일부로서 기록한다. 증거이미지를 Case 파일에 추가할 경우, CRC 값과 Hash 값을 통해 변경여부를 확인한다.



5-3. 압축 (Compression)

Encase는 압축기술로 데이터를 상대적으로 작은 크기로 저장한다. 평균 50%의 압축률을 보이는 산업표준 알고리즘을 이용하며, 디스크의 대부분이 Text인 경우, 그 압축 비율을 보다 높아진다. 그러나 JPG 파일과 같이 이미 압축된 파일들이 많은 경우에는 그 효율이 낮다. 압축을 위한 시간이 더 소모되긴 하지만, 압축과정은 증거이미지에 전혀 영향을 끼치지 않으며, 압축 블록은 비압축 블록과 동일한 과정으로 검증된다. 즉, 서로의 Hash Value는 동일한 값을 가진다.

5-4. Case 파일

Case 파일은 하나의 Case에 대한 구체적인 정보(하나 또는 그 이상의 증거이미지에 대한 포인터, 북마크, 검색 결과, 정렬, 해쉬 분석 결과, Signature 분석 결과 등)를 포함한다. Encase를 통해 증거이미지를 분석하기 위해서는 그 이전에 Case 파일이 생성되어야 한다.

5-5. Case 백업 파일

백업 파일은 Encase가 설치된 폴더의 Backup이라는 하위폴더 내에 자동으로 기본 10분마다 저장되며, .CBAK 이라는 확장자를 지닌다. .CASE 파일에 에러가 발생한 경우, .CBAK 파일이 열릴 수 있으며, .CBAK 파일을 .CASE 파일로 바꿀 수 있다.

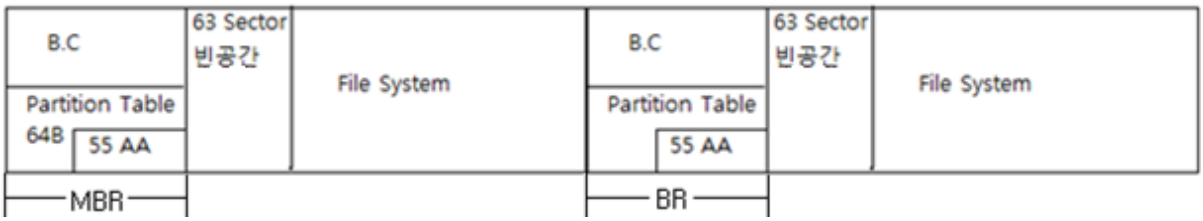
5-6. 환경 설정 파일

Encase 환경설정 파일은 일련의 초기화 파일(.INI)들이라고 할 수 있다. Signature 테이블, 파일 유형, 파일 viewer, 필터, Global Keyword 등 Global setting 정보들은 모든 Case와 모든 증거이미지에 적용된다. 일반적으로 Encase 설치 폴더의 Config라는 하위폴더에 위치한다.

6. 파티션 복구 (Partition Recovery)

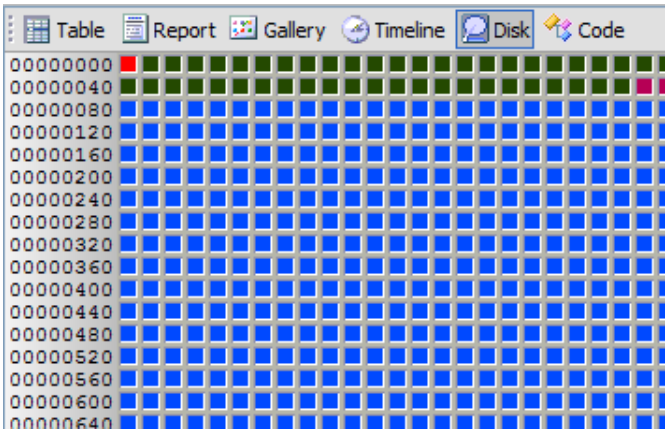
Physical disk를 조사할 때 그 첫번째 단계는 그 전체 크기를 계산해보는 것이다. 예를 들어, 250-GB의 Physical disk로부터 생성한 증거이미지로 조사를 하는데, 이 디스크에 하나의 100-GB의 파티션만 존재한다고 나타난다면, 150-GB의 공간은 계산되지 않은 것이다. 만약 이 공간이 삭제된 파티션을 포함하고 있다면, 조사관은 이 파티션을 복구하여 유용한 정보들을 추출할 수 있을 것이다.

6-0. 볼륨 분석 (Volume Analysis) – Volume Layout 파악



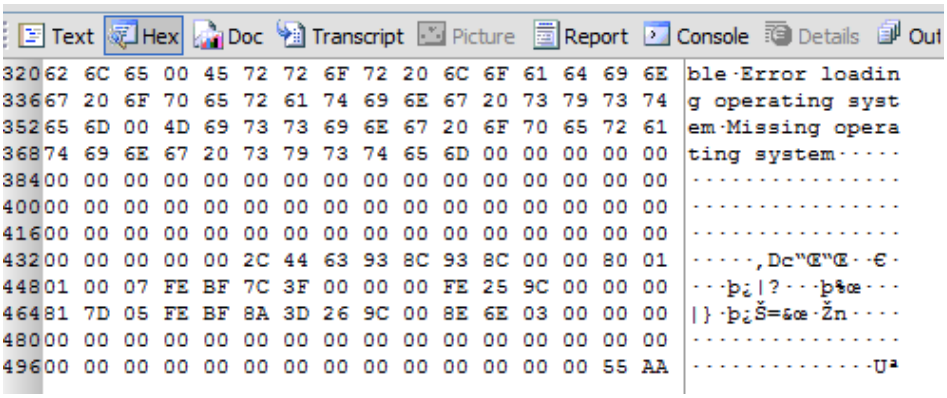
0. 우선 MBR을 찾은 다음 partition의 합이 맞는지 점검

(숨어있는 파티션 -저장되지 않은 파티션- 이 있는지 확인하는 절차이다.)



// MBR : 적색으로 표시된 Sector

MBR은 Master Boot Record의 약자로, Hard drive의 첫 번째 Physical sector (sector 0)로서, 512 bytes의 크기를 가진다. Executable Code (Active Partition의 Volume Boot Record를 위치시키고 로드함), 에러 메시지, 마스터 파티션 테이블과 마지막의 55h AAh의 Signature로 이루어진다.



// MBR 의 510, 511 번째 byte 가 55 AA 라는 것을 확인

Partition

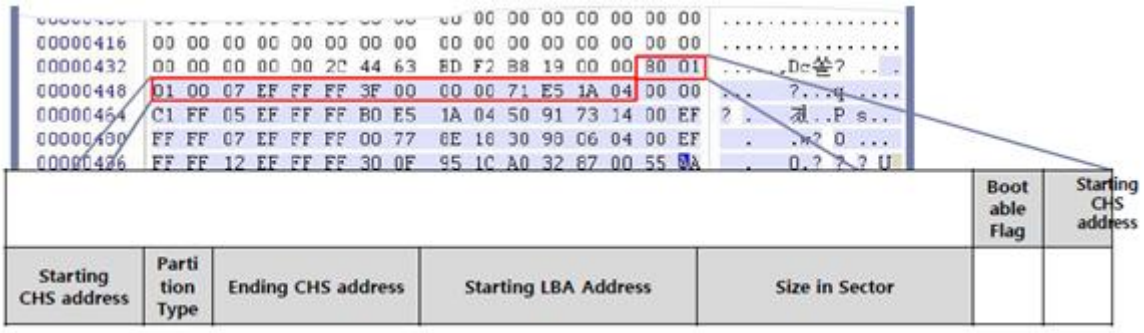
Name	Id	Type	Start Sector	Total Sectors	Size
	07	NTFS	0	10,233,405	4.9GB

Name	김보안
Description	Physical Disk, 10,485,216 Sectors 5GB
Logical Size	0
Initialized Size	0

// 위의 분석 결과, 파티션 크기의 합과 전체 크기가 불일치.
(안 쓰는 공간이 있거나 기존 사용 한 후 삭제한 파티션이 있는 경우, 혹은 파티션 테이블이 깨졌거나 예전에 포맷하고 남은 영역으로 의심할 수 있다.)

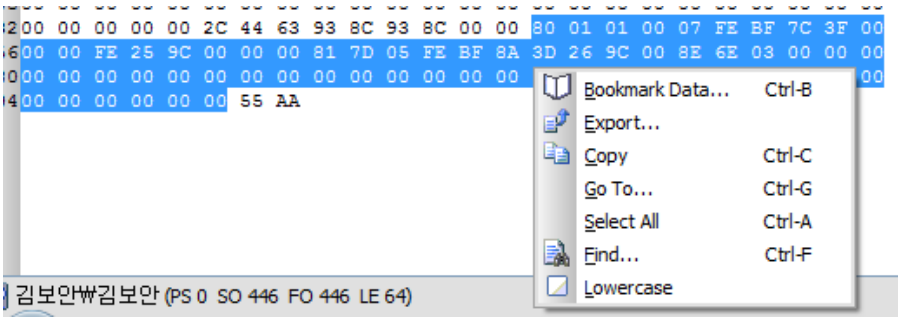
1. Partition Table 해석 (Entry size : 16 bytes, 최대 4개의 Entry)

- **Partition Table Layout**

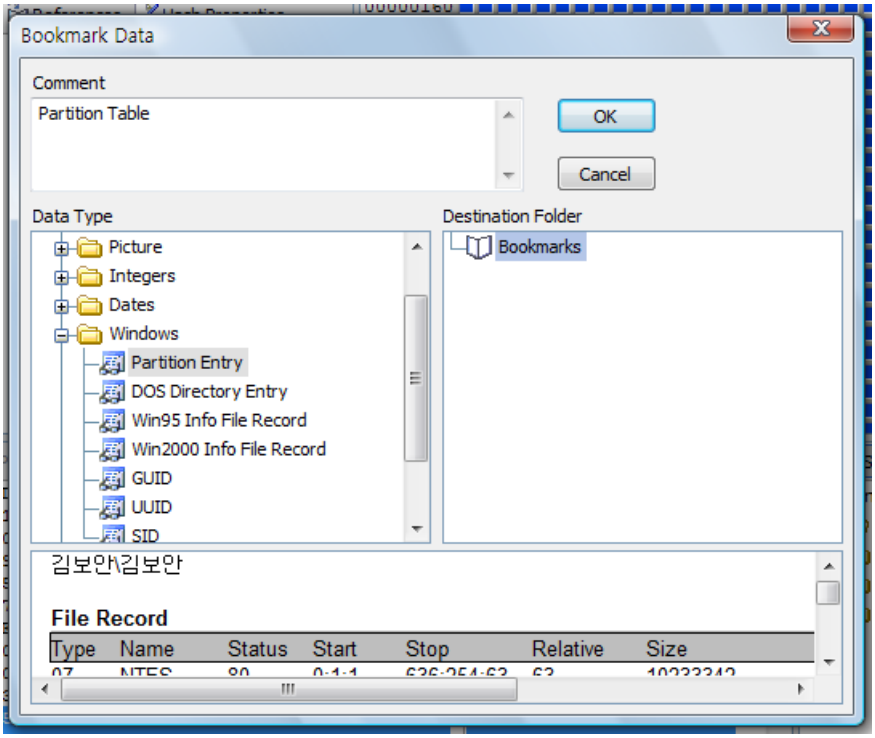


// Bootable Flag : Active (1 byte, 80 = YES, 00 = NO)
Starting CHS address (3 bytes, 단위 : sector)
Partition type (1 byte)
Ending CHS address (3 bytes, 단위 : sector)
Starting LEA Address : Relative Sector offset (4 bytes, 단위 : sector)
Size in Sector : Total Sectors of partition (4 bytes)

- **Bookmark 기능** : 데이터를 분석하다가 나중에도 다시 보아야 할 데이터를 표시해두는 기능인데, 이 과정 중에 데이터를 해석하게 할 수 있다.



// MBR 의 Signature 앞 64byte 가 바로 Partition Table 이다.
여기에서 Bookmark 는 마우스 오른쪽 버튼 클릭 -> Bookmark Data



// Encase의 기본 템플릿 중 Windows - Partition Entry를 선택하면 분석결과가 Bookmark된다. (Data Filetype : Partition Entry)

2. Partition 위치 확인

Bookmarks

Case Time Settings

Account for seasonal Daylight Saving Time	Yes
Convert all dates to correspond to one time zone	No

1) 김보안\김보안
Partition Table

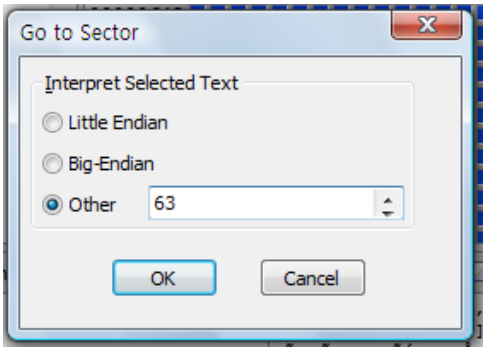
File Record

Type	Name	Status	Start	Stop	Relative	Size
07	NTFS	80	0:1:1	636:254:63	63	10233342
05	Ext DOS	00	637:0:1	650:254:63	10233405	224910
00	None	00	0:0:0	0:0:0	0	0
00	None	00	0:0:0	0:0:0	0	0

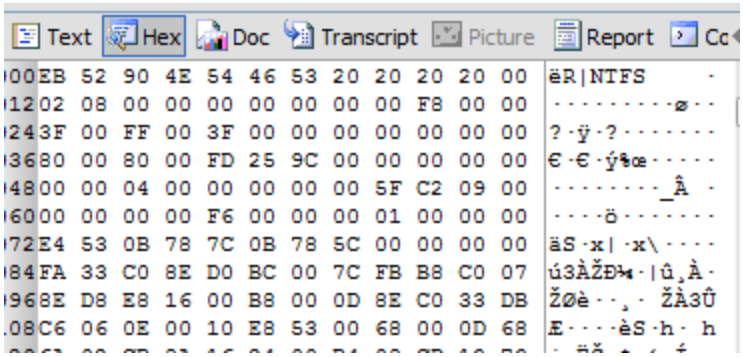
// Size값을 통해 Partition 위치 확인

3. 첫 번째 파티션 확인

- NTFS 가 시작하는 곳은 MBR 로부터 63 sector 만큼 떨어져 있는 곳이다.
  (File System 의 시작인 VBR 은 MBR 로부터 63 sector 떨어진 곳에 위치.)



// Ctrl + G (Go to Sector : 63)

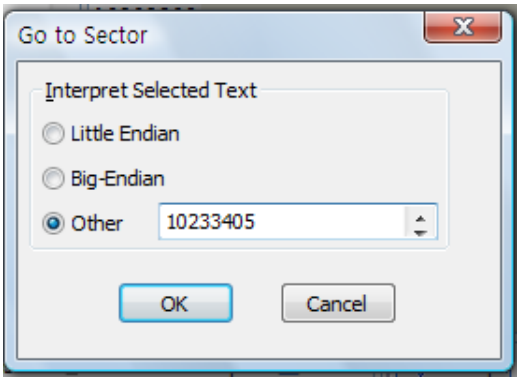


// VBR에서 File System Signature 확인 (OEM String : NTFS)

VBR은 Volume Boot Record의 약자로서, 각 File System마다 맨 앞의 Sector에 위치하여 파티션의 크기(VBR의 Sector Offset 40에 위치하며, Little Endian으로 저장되어 있으므로 우클릭 후 디코딩하여 확인가능) 등의 정보가 들어있다. MBR의 Partition Table에 저장된 파티션 크기 정보보다 VBR에 저장되어 있는 파티션 크기 정보가 1 sector 모자라는 것을 확인할 수 있는데, 이는 VBR의 백업본을 해당 파티션의 마지막 섹터에 보존하는 NTFS의 포맷 특성으로부터 비롯된다.

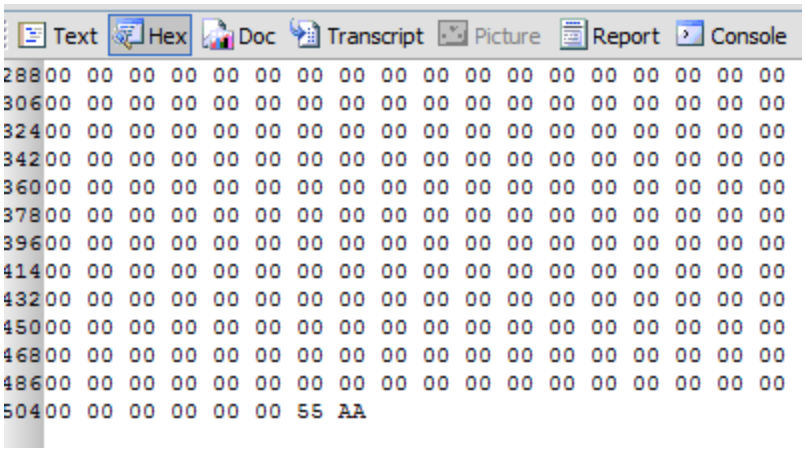
4. 다음 파티션 확인

- Ext DOS 가 시작하는 곳은 NTFS 파티션의 Size 이후이다.
  NTFS 파티션이 10,233,405 라고 사용한다고 하였으나 Sector 가 0 번으로 시작하므로 새로운 파티션은 10,233,405 부터 사용되는 것이다.

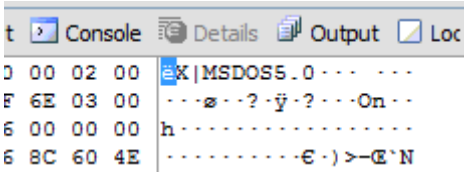


// Ctrl + G (Go to Sector : 10233405)

- 이 경우 MBR 이 아닌 보통의 Boot Record 임을 확인하였다. 따라서 Signature 를 제외한 나머지 부분은 모두 빈 공간인 것이 정상이다.



- BR 로부터 63 sector 떨어진 곳에 파일시스템의 Signature 가 존재한다.
  (Partition Table 을 분석한 파일시스템과 해당 파일시스템의 Signature 가 불일치하는 경우, 의심스러우므로 다시 확인해보아야 한다.)



6-1. 파티션 복구 원리

0. Unallocated Cluster

- Encase 가 제대로 파일 시스템을 인식하지 못한 상태

1. MBR 의 복원

- Bootcode 는 깨끗한 MBR 의 Bootcode 를 복사하여 붙여넣고,
- Partition Table 이 깨진 경우 각각의 파티션 정보를 수집하여 직접 Partition Table Layout 에 맞춰 입력해주도록 한다.
- Forensics 관점에서는 MBR 을 완전히 복원할 필요 없이(부팅이 목적이 아니므로), Partition Table 만 복원하는 것으로 충분하다.

2. VBR 의 복원

- VBR 은 수작업으로 복원하기가 어렵다. 각 필드 하나하나를 채우기가 어렵기 때문이다.
- 일반적인 경우 백업본을 활용하는 것이 좋다.
- NTFS 인 경우 파일 시스템의 맨 마지막 섹터에 백업본이 있다. (파티션 매직 같은 프로그램을 사용했을 경우 백업본을 정상적으로 놔두지 않은 경우가 많아 복원이 어렵기도 하다.)
- FAT 인 경우 FAT12 나 FAT16 은 백업본이 없고, FAT32 의 경우 백업본이 존재하나 위치가 고정되어 있지 않다. (아이러니하게도 VBR 에 백업본의 위치가 기록되어 있다.)
- 따라서, VBR 이 손상되었을 경우, 문자열 Search 를 하거나(하지만 결국 Reserved 영역 안에 있으므로, 섹터 사이즈가 크지 않기 때문에 시간은 얼마 걸리지 않는다.), MS 의 경우에 대체로 VBR 로부터 6 섹터만큼 떨어진 곳에 있으므로 이를 확인해본다.
- Encase Forensic Training 버전의 경우, 복구할 VBR 위치에서 'Add Partition' Option 중 'Backup Boot Record'를 체크하면 자동으로 VBR 백업본을 활용하여 Partition 을 인식하는 기능을 가진다.
- Encase LE(Law Enforcement) 버전의 경우에는 위의 Option 이 존재하지 않으므로, 수작업으로 직접 백업본을 원래 VBR 에 덮어써주어야 한다.
- 그러나 Encase 는 이미지의 직접 Data 수정이 불가하므로, Encase Prosuite(인증서 필요) 중 PDE 기능을 활용하여 실제 디스크로 인식시키면 된다. 이 또한 없는 경우, 해당 이미지 파일을 Winhex 에서 열어 백업본 위치의 Block 을 복원할 위치에 복사하면 된다.

c.f) Encase Prosuite

- PDE(Physical Disk Emulator, 실제 디스크로 인식시켜서 활용가능),
- VFS(Virtual File System),
- EDS(Encase Decryption Suite) 기능 활성화.

c.f) PDE 활용

- Mount as Emulated Disk : Encase 가 Server 역할을 하고, Client 에서 수정여부 기록 가능 (Disable Caching uncheck)

3. Encase 가 인식한 파티션 외에 다른 파티션이 존재하는지 여부 확인 (Keyword Searching)

- 하드 전체를 선택 한 다음 Unused 공간의 Report 를 확인하여, 공간이 너무 많이 남은 경우 다른 파티션이 있었는데 삭제 된 경우로 의심해 볼 수 있다. (또는 MBR 손상을 의심할 수 있다.)
- 이 때 Unused space 에서 'OEM String'을 Keyword Searching 을 통해 VBR 을 찾아서 확인해 볼 수 있다.
- Keyword Searching 방법

  New Keyword : 키워드 생성
  GREP (check) : 정규식 사용
  Search expression : (NTFS)|(MSWIN4.1)|(MSDOS5.0)
  Name : VBR OEM String
  Case Sensitive (check) : 대소문자 구별하는 경우
  Unused Disk Area (check) : 인식하지 못한 디스크 공간에서 VBR 검색
  Selected entries only – Search each entry for keywords.
  Selected keywords only (check) : 선택한 키워드만 검색

- 결과는 'Search Hits' 탭에 나오며. Backup Boot Record 여부를 섹터 번호를 확인한다. Encase 는 Sector 단위로 검색을 하지 않아 오탐이 많으므로 주의한다.
- 이후 VBR 시작 위치가 확인되면, 직접 해당 위치에 가서 Add partition 을 해주면 된다.
  (이때 Unused sectors before VBR 에서 Primary Partition 의 경우, 앞에 Boot Record 가 없이 바로 붙어있으므로 '0', Extended Partition 의 경우, VBR 의 63 섹터 앞에 Boot Record 가 존재하므로 '63'을 입력한다.)

c.f) Keyword 는 Global Keyword 와 Local Keyword 로 나뉜다.
Global Keyword : Case 탭 옆의 Keyword. 모든 케이스에서 사용가능.
Local Keyword : Home 옆의 Keyword. 해당 케이스에서만 사용가능.

6-2. 파티션 복구 실습 (4 Cases)

① Boot Record의 VBR위치 정보 손상된 경우

- **복원 전 MBR : Partition Table Layout**

Bootable Flag	Starting CHS	Partition Type	Ending CHS	Starting LBA	Size in sector
00	01 01 00	07	FE 3F 07	3F 00 00 00	C9 F5 01 00
00	00 01 08	05	FE 3F 0D	08 F6 01 00	86 78 01 00

- **하드 전체 Layout**

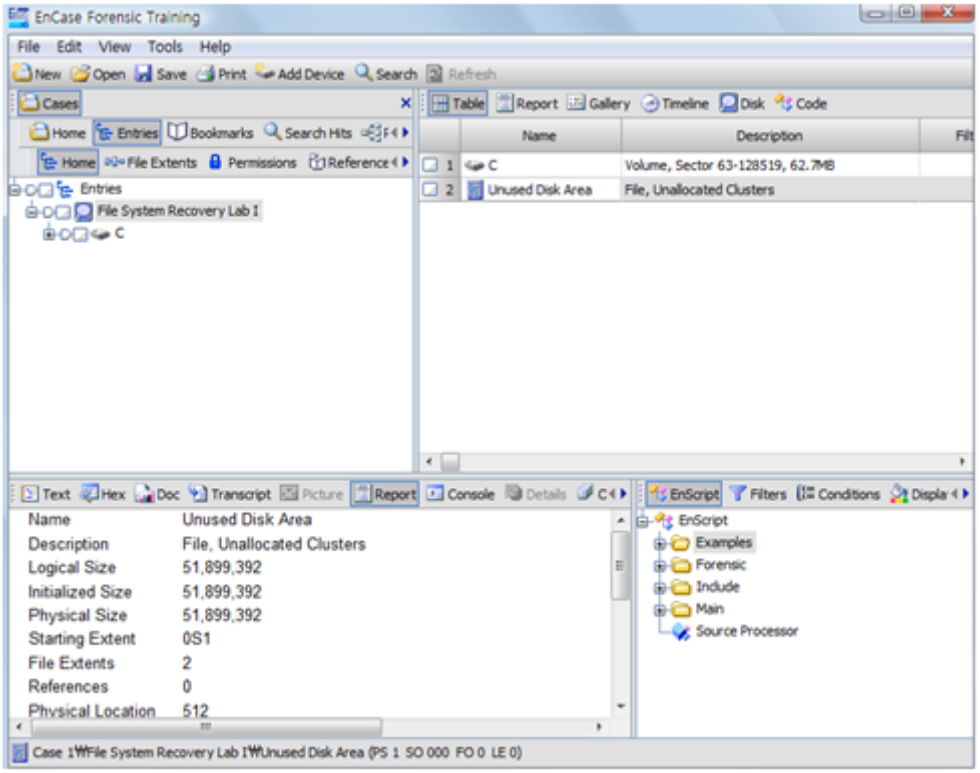
MBR		Primary 1 (NTFS)	BR		Extended 1 (FAT32)

- **하드디스크 정보 (229,824 Sectors, 112.2MB)**

Type	Name	Status	Start	Stop	Relative	size
07	NTFS	00	000101	073FFE	63	128457
05	FAT32	00	080100	0D3FFE	128520	96390

- **손상여부 확인**

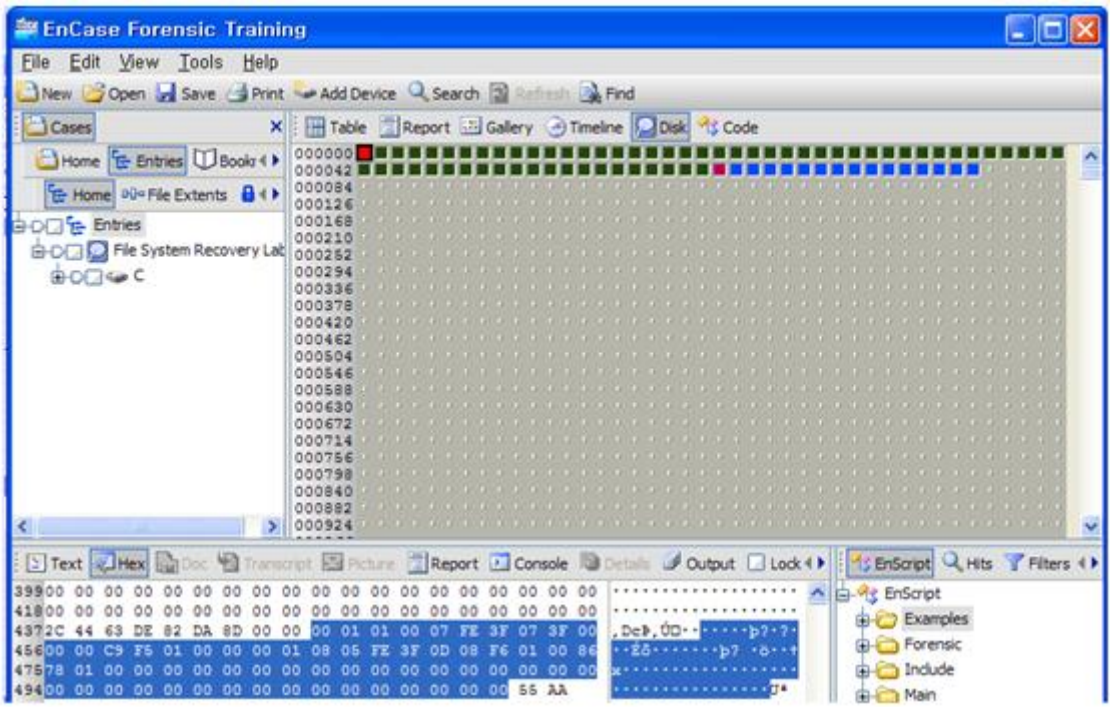
a. Unused Disk Area 의 크기 확인



// 약 50 MB 로 나타나, 파티션이 삭제되거나 인식되지 않았을 가능성 있음

b. MBR 확인

Partition Table 확인 결과 총 2 개의 Partition 정보가 수록되어 있으나, Encase 에서는 C 드라이브 1 개만 인식하고 있음

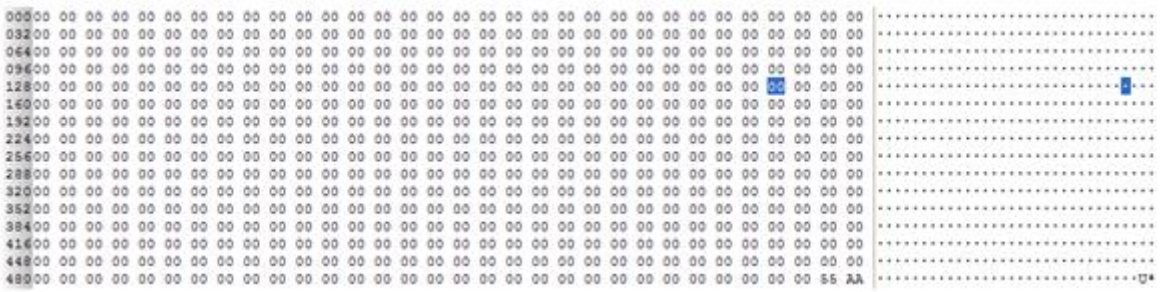


// Partition Table 해석 : 총 229824 섹터 중 첫 번째 파티션은 128457 개의 섹터의 용량 차지, 시작 LBA 는 63 섹터. 두 번째 파티션은 96390 섹터의 용량을 차지, 시작 LBA 는 128458 섹터.

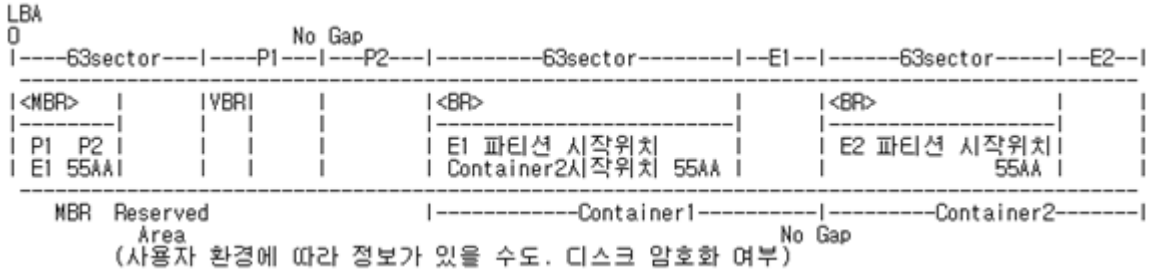
- **인식 오류 발생 원인**

두 파티션 모두 Master Boot Record 의 Partition Table 에는 제대로 정보가 나타나 있었고, 첫 번째 파티션 (Primary Partition)은 VBR 또한 이상 없었지만, 두 번째 파티션의 경우 Extended Partition 으로 보이는데, BR 에 VBR 위치 정보가 기록되어 있지 않아(손상) 인식이 되지 않음.

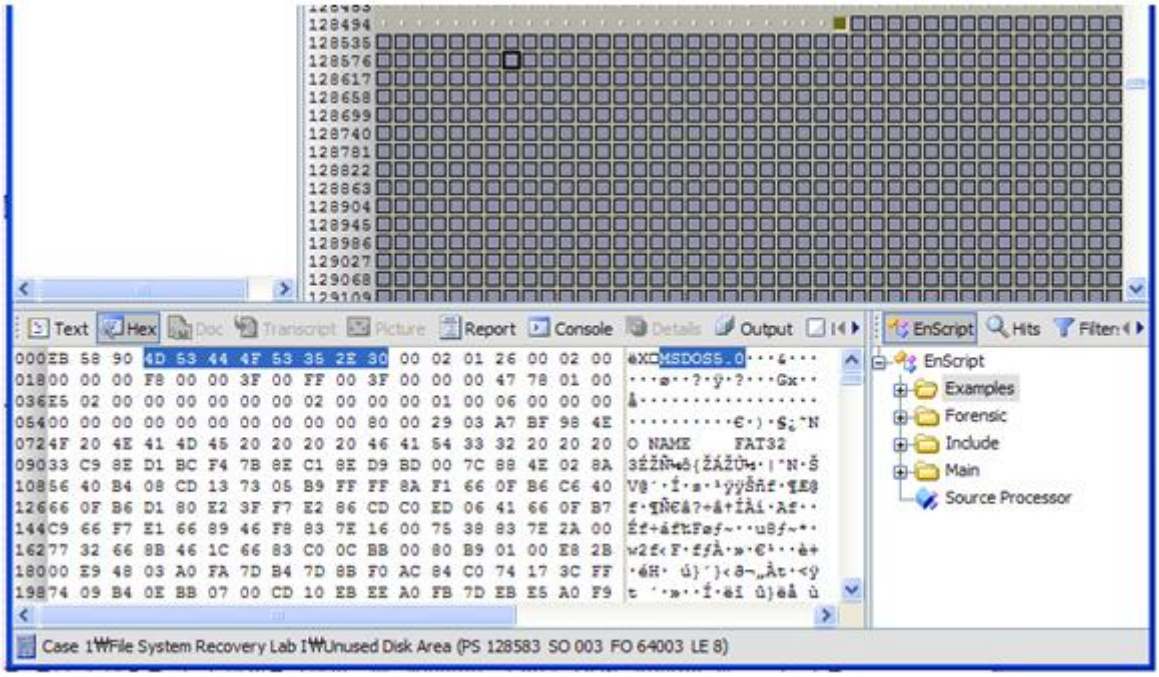
- **복원 과정**
  a. MBR 정보 확인하여 두 번째 파티션 시작위치(Starting LBA 값 08 F6 01 00 : 128454 sector)로 이동
  b. 마지막 2 Bytes 에 "55 AA" Signature 뿐으로, OEM String 이 없음
    b-1. 2 번째 파티션이 Primary partition 인 경우, 이 섹터는 VBR 로서 OEM String 이 삭제되고 File System 의 Meta Data 등이 삭제된 것임
    b-2. 2 번째 파티션이 Extended partition 인 경우, 이 섹터는 BR 로서 이 파티션의 시작위치(다음 Container 가 존재하는 경우, 다음 Container 의 시작위치도 존재)를 가리키는 Partition Table 이 삭제된 것임



c.f) Partition Layout (Primary Partition 2 개, Extended Partition 2 개인 경우)

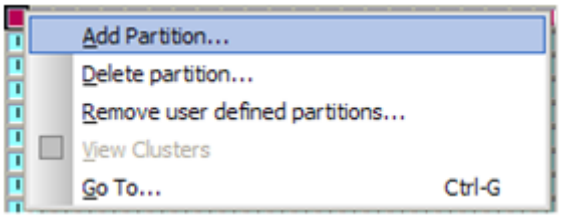


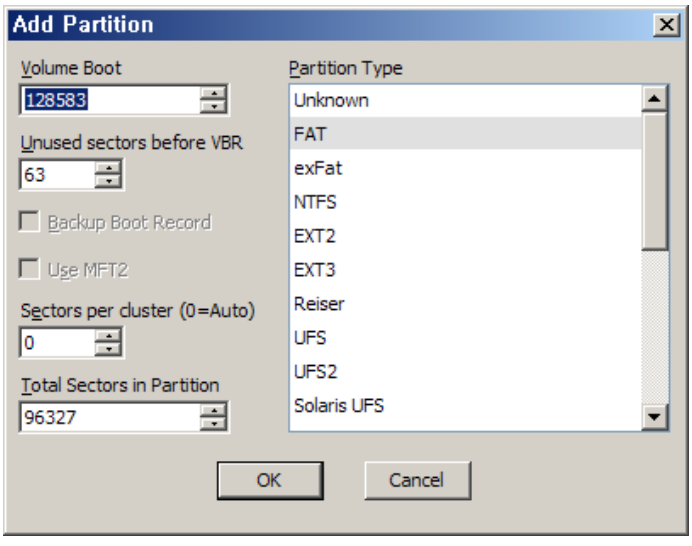
c. Extended Partition 인지 여부를 알기 위해 63 sector 이후 (125883 sector) 확인



// OEM String 이 MSDOS5.0 이므로 파일 시스템의 유형은 FAT32 로 확인

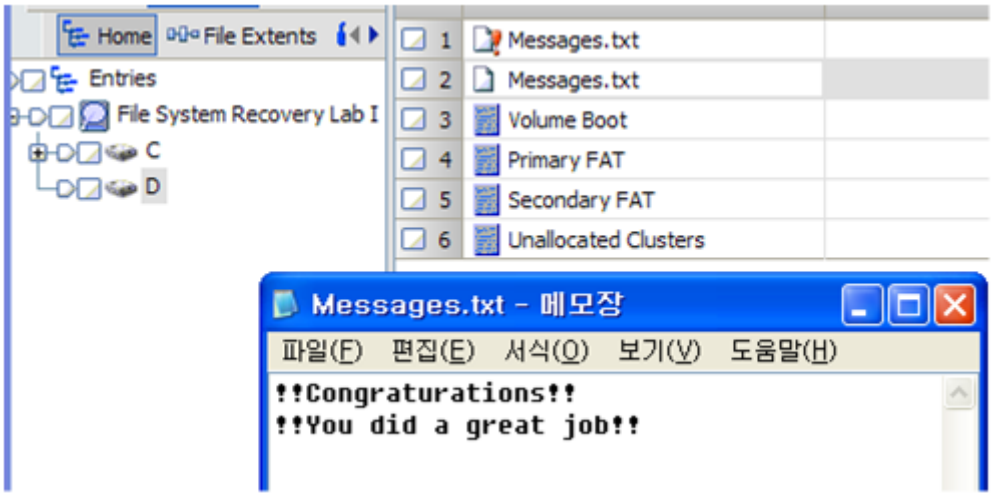
d. 2 번째 파티션의 시작지점(VBR)인 128583sector 에서 새로운 파티션 추가(Add Partition)





// 확장 파티션이므로 'Unused Sectors before VBR = 63'

- **파티션 복원 완료** : 기존의 C 드라이브 외에 추가로 D 드라이브 생성



② MBR의 Partition Table 일부 및 확장 파티션의 BR이 손상된 경우

- **복원 전 MBR : Partition Table Layout**

Bootable Flag	Starting CHS	Partition Type	Ending CHS	Starting LBA	Size in sector
00	01 01 00	07	FE 3F 07	3F 00 00 00	C9 F5 01 00

- **하드 전체 Layout**

MBR		Primary 1	BR		Extended 1

- **하드디스크 정보 (229,824 Sectors, 112.2MB)**

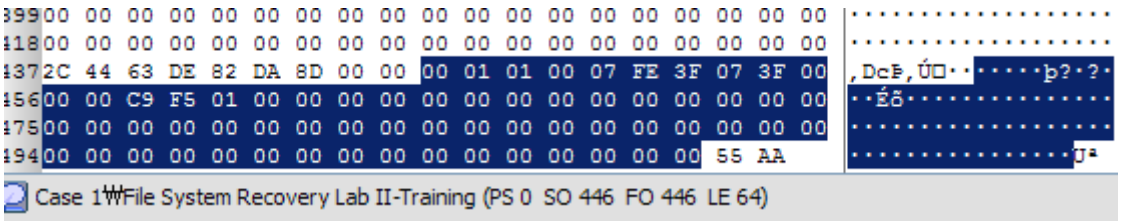
Type	Name	Status	Start	Stop	Relative	size
07	NTFS	00	000101	073FFE	63	128457
05	FAT32	00	080100	0D3FFE	128520	96390

- **손상여부 확인**
  a. Disk 전체 섹터와 인식된 파티션의 섹터 크기 확인
     전체 섹터 크기 : 229824 sector
     첫 번째 파티션 크기 : 128457 sector
  b. 전체 섹터 중에서 거의 반만 가진 하나의 파티션만을 가졌다고 하기는 의심스럽다. 첫 번째 파티션 후에도 새로운 파티션이 있는지 찾아봐야 한다.

Total Size	117,669,888 Bytes (112.2MB)
Total Sectors	229,824
Disk Signature	DE82DA8D
Partitions	Valid

- **인식 오류 발생 원인**

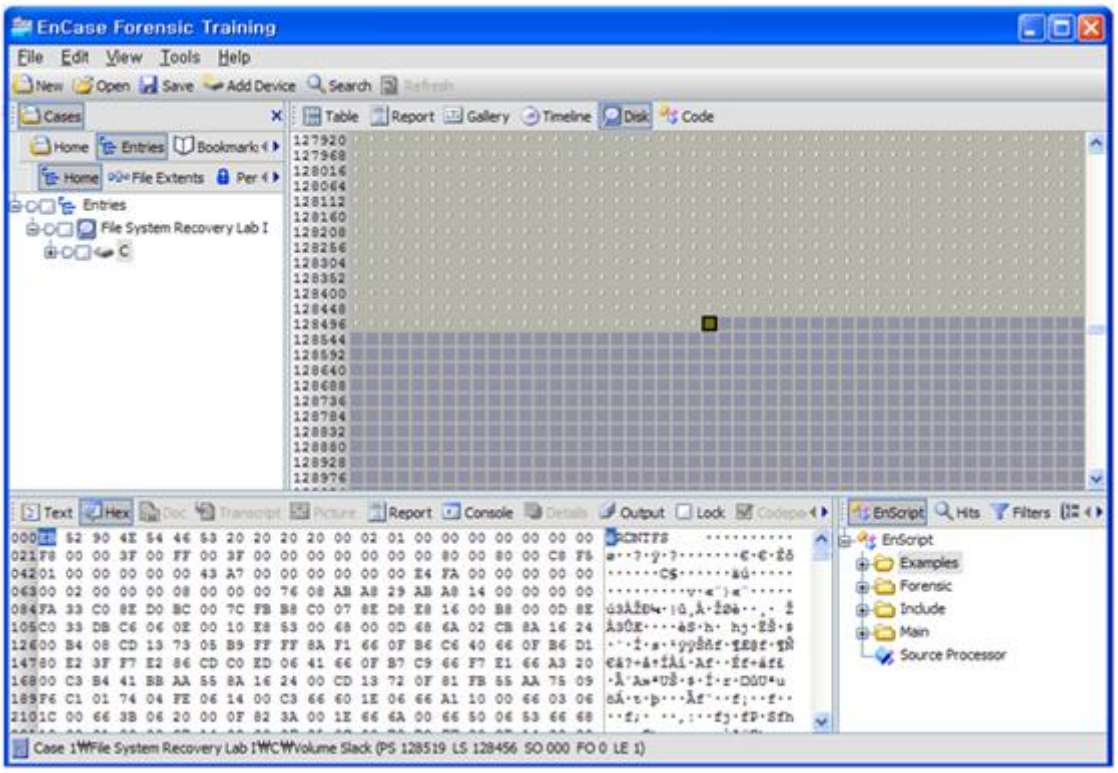
  MBR 에는 파티션 하나만 기록이 되어 있었으나, Unused Disk Space 가 약 52MB 로 나타나, 한 개의 파티션이 더 있을 것을 의심. 첫 번째 파티션의 끝 지점을 확인하였더니 다음 섹터(128520)의 마지막 부분에 55 AA 의 Signature 가 확인됨. 그로부터 63 섹터 떨어진 128583 섹터를 확인하였더니 VBR 로 판단됨. 즉, MBR 손상 및 확장 파티션의 BR 손상.



- **복원 과정**

  a. 첫 번째 파티션은 NTFS 임을 확인(Signature : OEM String – NTFS)

  b. 마지막 섹터에 VBR 백업본 확인



  c. 다음 섹터를 확인

  c-1. MBR 정보 확인한 두 번째 파티션 시작위치 : Starting LBA 값 08 F6 01 00 와 일치

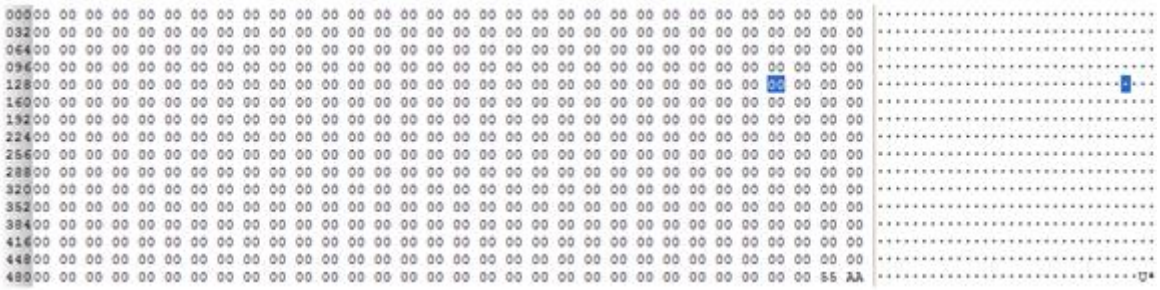
c-2. 새로운 파티션이 시작되어야 함.

MBR 확인 결과 2 번째 파티션 시작주소(128520 섹터)

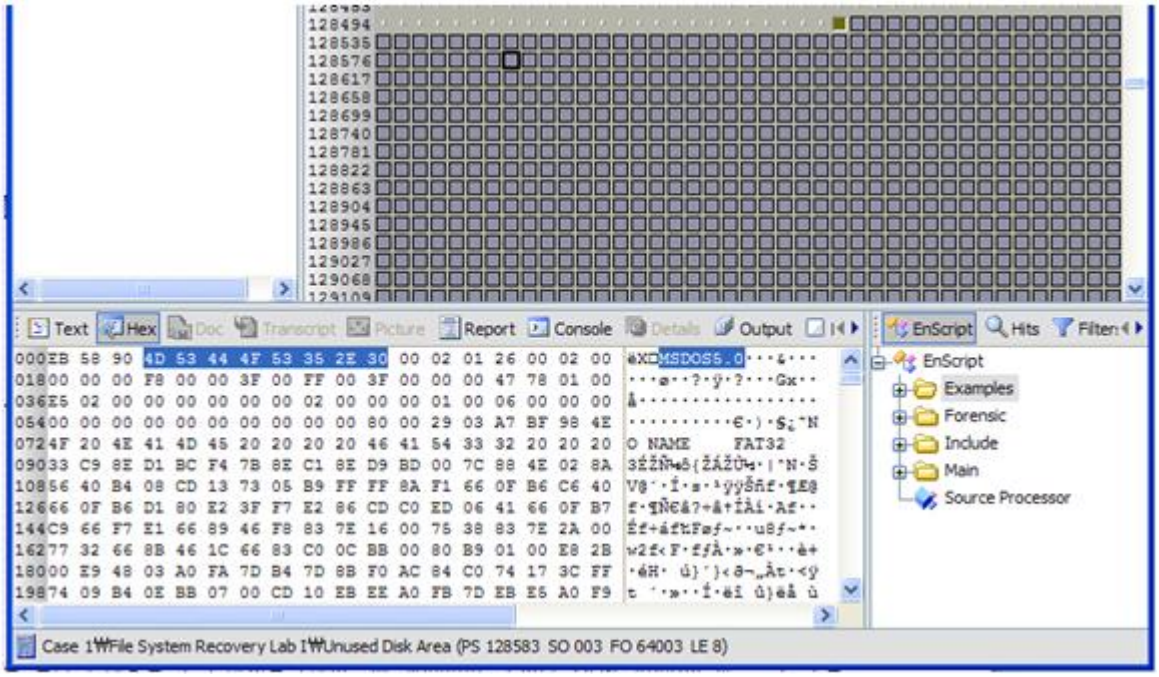
d. 마지막 2 Bytes 에 "55 AA" Signature 뿐으로, OEM String 없음

d-1. 2 번째 파티션이 Primary partition 인 경우, 이 섹터는 VBR 로서 OEM String 이 삭제되고 File System 의 Meta Data 등이 삭제된 것임

d-2. 2 번째 파티션이 Extended partition 인 경우, 이 섹터는 BR 로서 이 파티션의 시작위치(다음 Container 가 존재하는 경우, 다음 Container 의 시작위치도 존재)를 가리키는 Partition Table 이 삭제된 것임

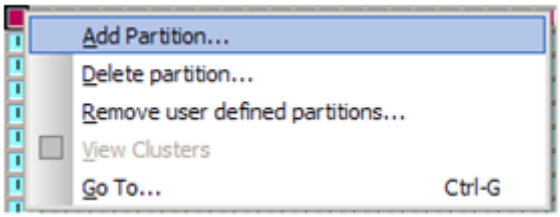


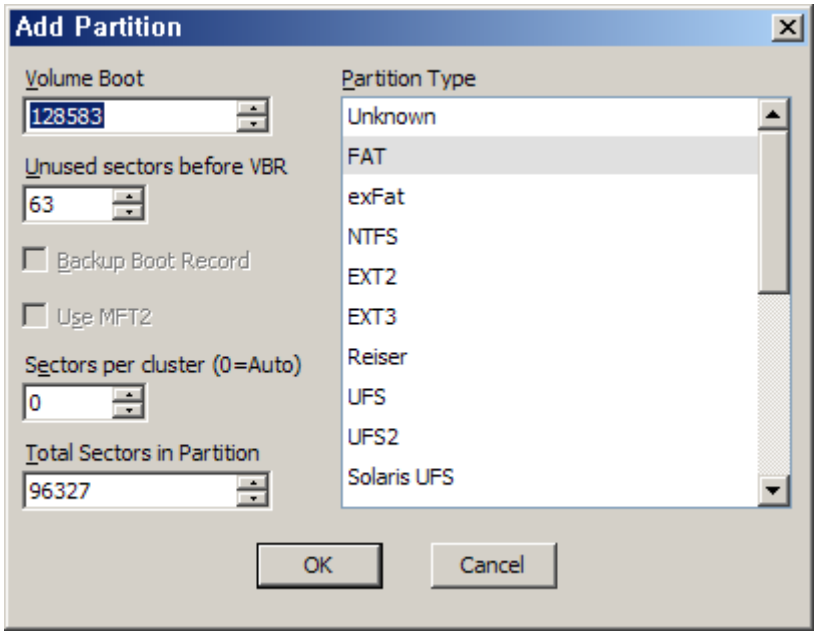
e. Extended Partition 인지 여부를 알기 위해 63 sector 이후 (125883 sector) 확인 :



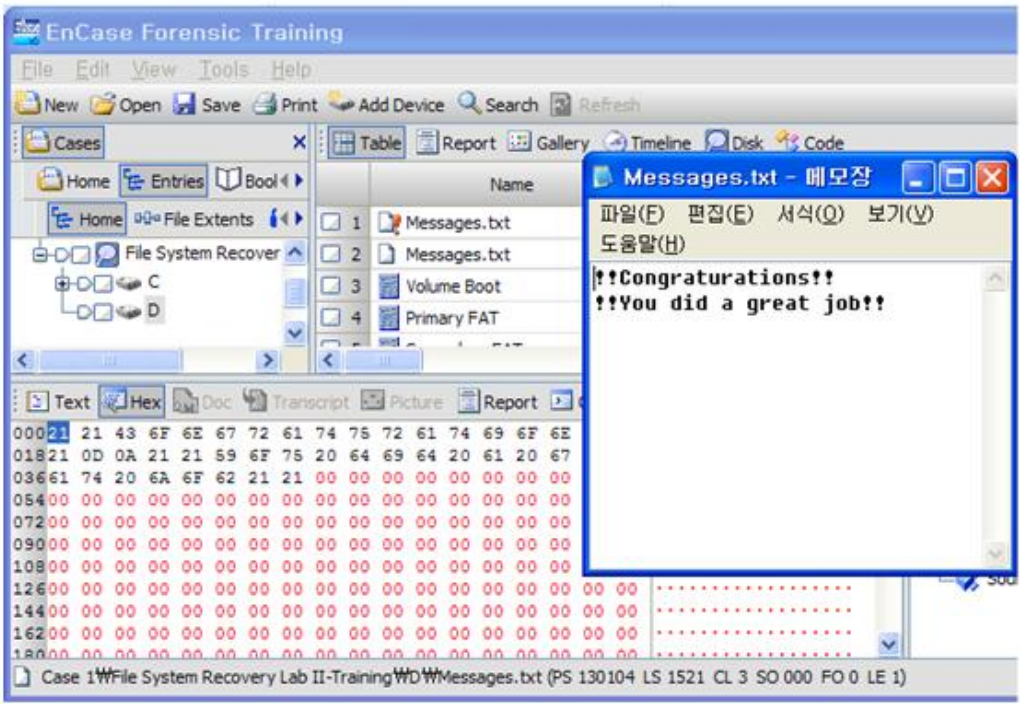
// OEM String 확인, FAT 확인(File System 의 Meta Data) : VBR Extended Partition 임, 즉 BR 정보가 조작된 것임

f. 2 번째 파티션의 시작지점(VBR)인 128583sector 에서 새로운 파티션 추가(Add Partition)





- **파티션 복원 완료 :** 기존의 C 드라이브 외에 추가로 D 드라이브 생성



③ MBR의 Partition Entry가 전혀 없는 경우

- **복원 전 MBR : Partition Table Layout**

Bootable Flag	Starting CHS	Partition Type	Ending CHS	Starting LBA	Size in sector

- **하드 전체 Layout**

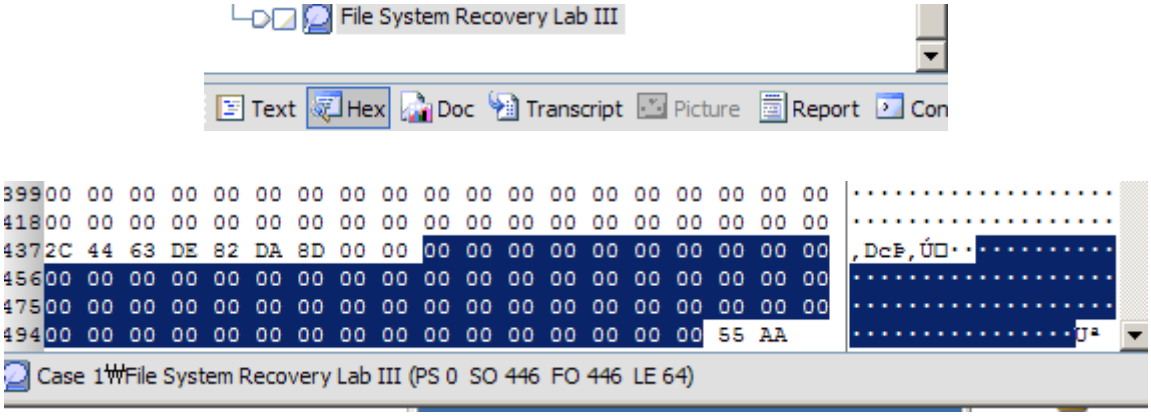
MBR		Primary 1	Primary 2

- **하드디스크 정보 (229,824 Sectors, 112.2MB)**

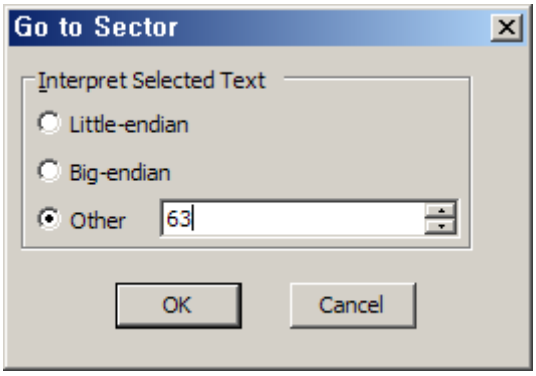
Type	Name	Status	Start	Stop	Relative	size
07	NTFS	00	000101	073FFE	63	128457
05	FAT32	00	080100	0D3FFE	128520	96390

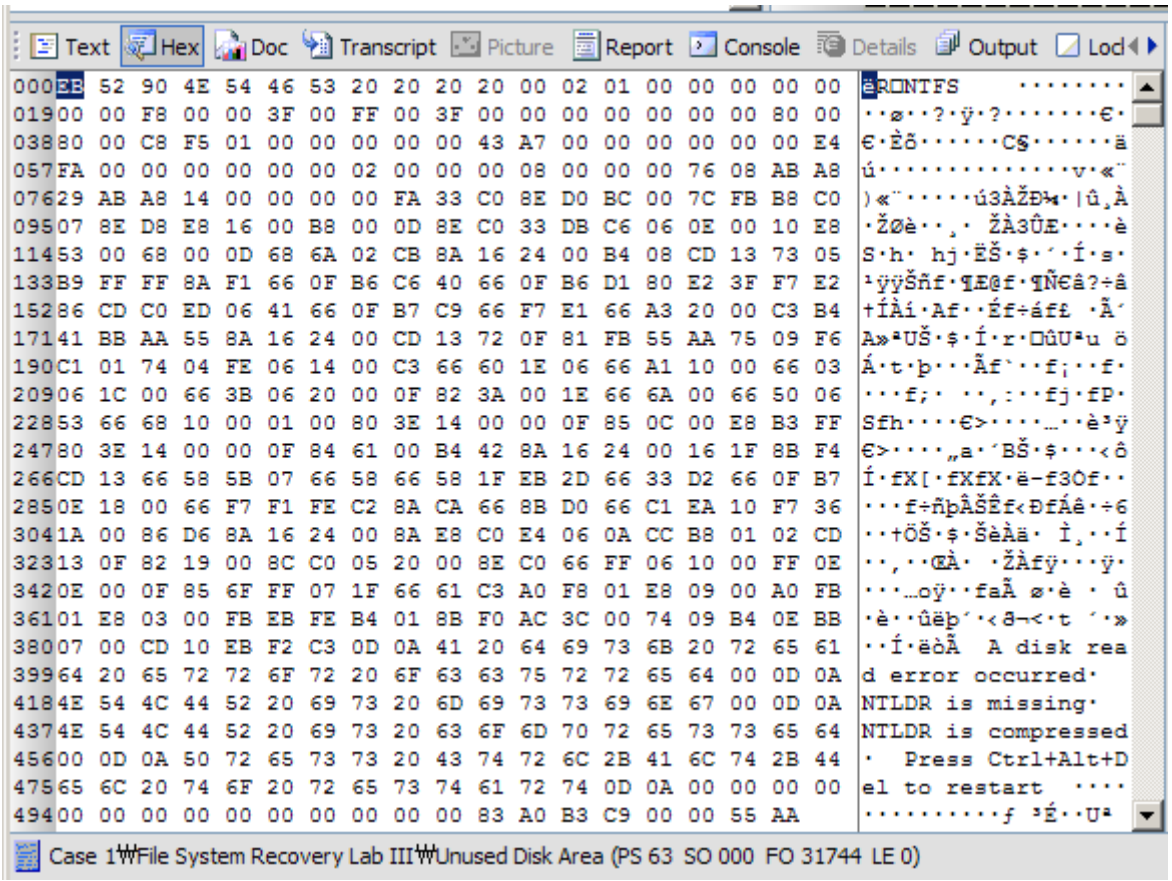
- **인식 오류 발생 원인**

  MBR 에 Partition Entry 가 전혀 기록되어 있지 않다. 즉, Encase 에서는 아무런 파티션도 인식하지 못하였다.



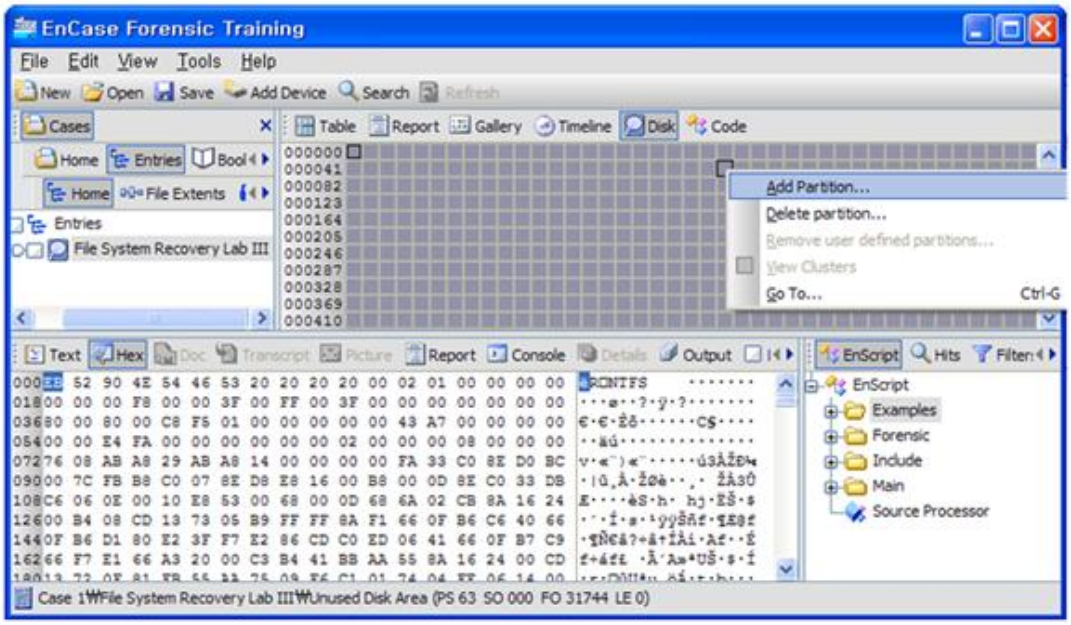
- **복원 과정**
  a. MBR 의 Partition Entry 를 고의적으로 삭제하였을 경우, 실제 파티션의 시작 지점에 위치하는 VBR 은 남아있는지를 우선적으로 확인한다. 이때, 첫 Primary Partition 은 63 번 섹터에서 시작하게 되므로 63 번 섹터를 확인.





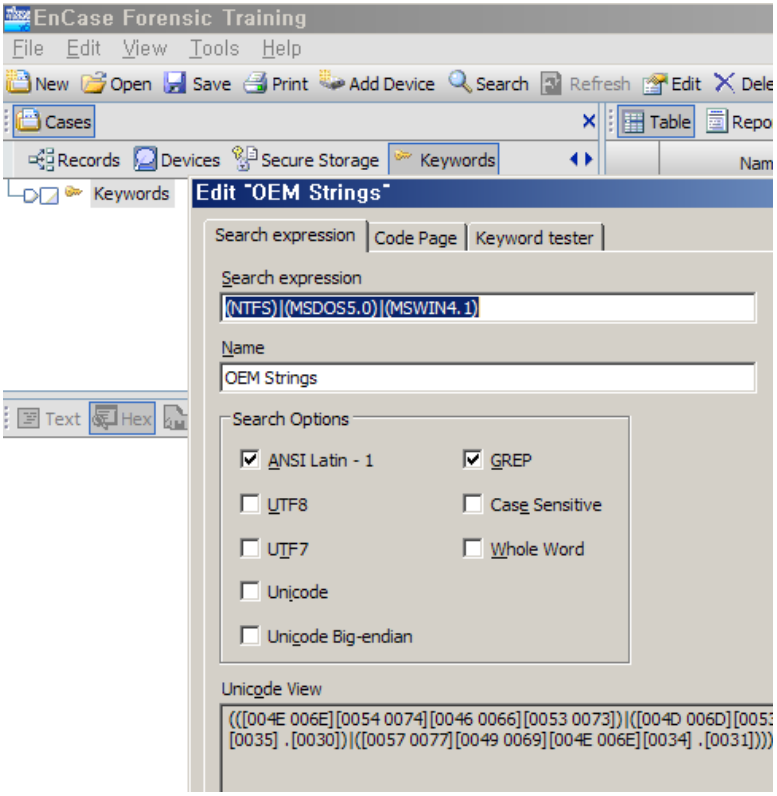
// OEM String 으로 보았을 때, NTFS 파일 시스템을 갖는 파티션이 존재함 확인

b. 첫번째 파티션(Primary, NTFS) 추가

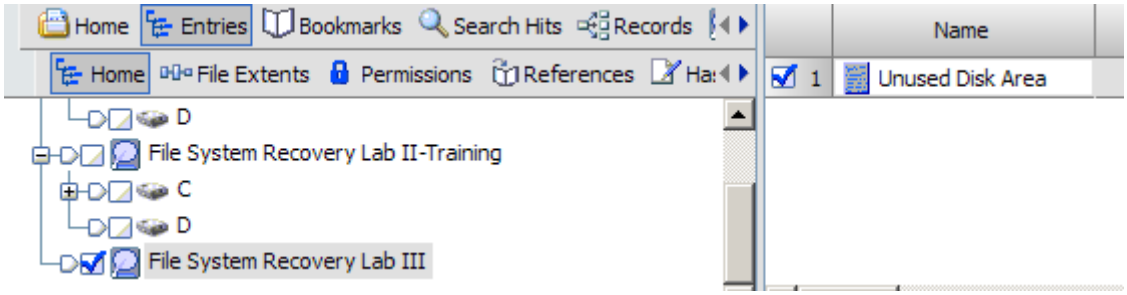


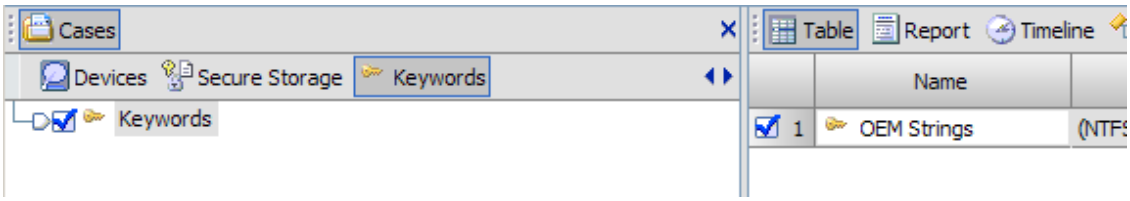
c. 하지만, 이때 파티션이 여러 개 존재하는지 여부 및 파티션의 크기와 시작 위치 등은 MBR 이 삭제 되었으므로 더 이상 알 수 없다.
따라서 다른 파티션을 찾아보기 위하여 Keyword 로 Search 한다.

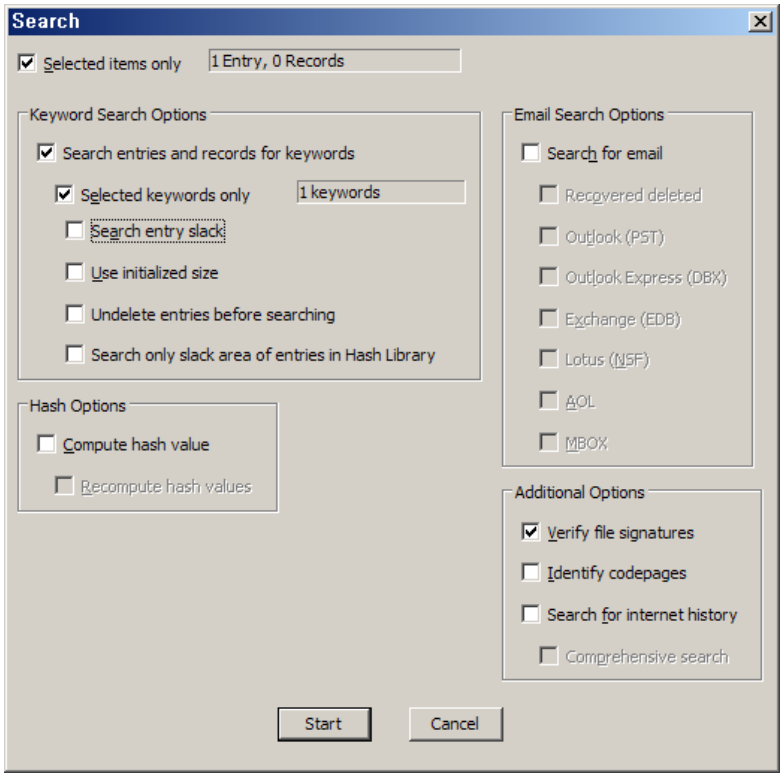
c-1. 먼저 OEM String 을 키워드로 등록한다.



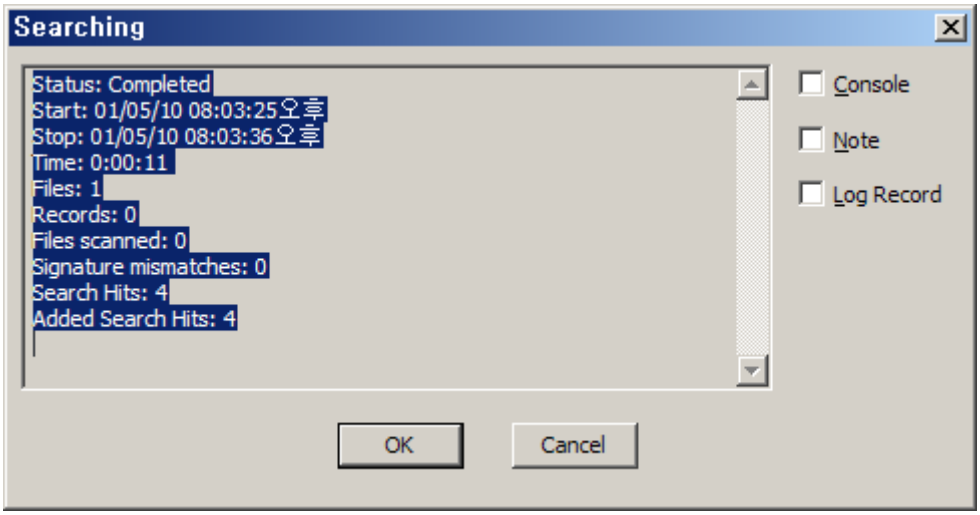
c-2. Unused Disk Area 와, OEM String 키워드를 체크하고 Search

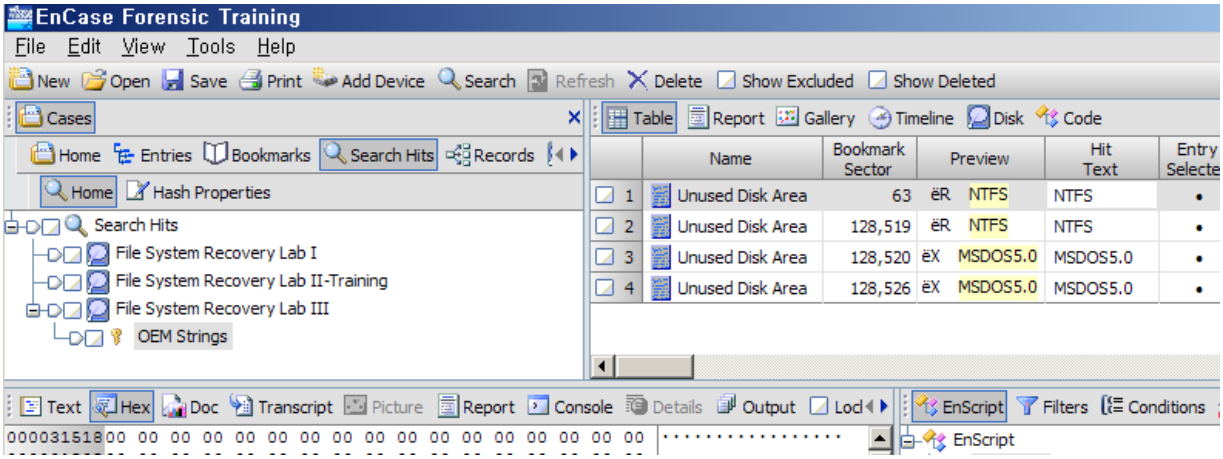






d. 검색 결과 확인



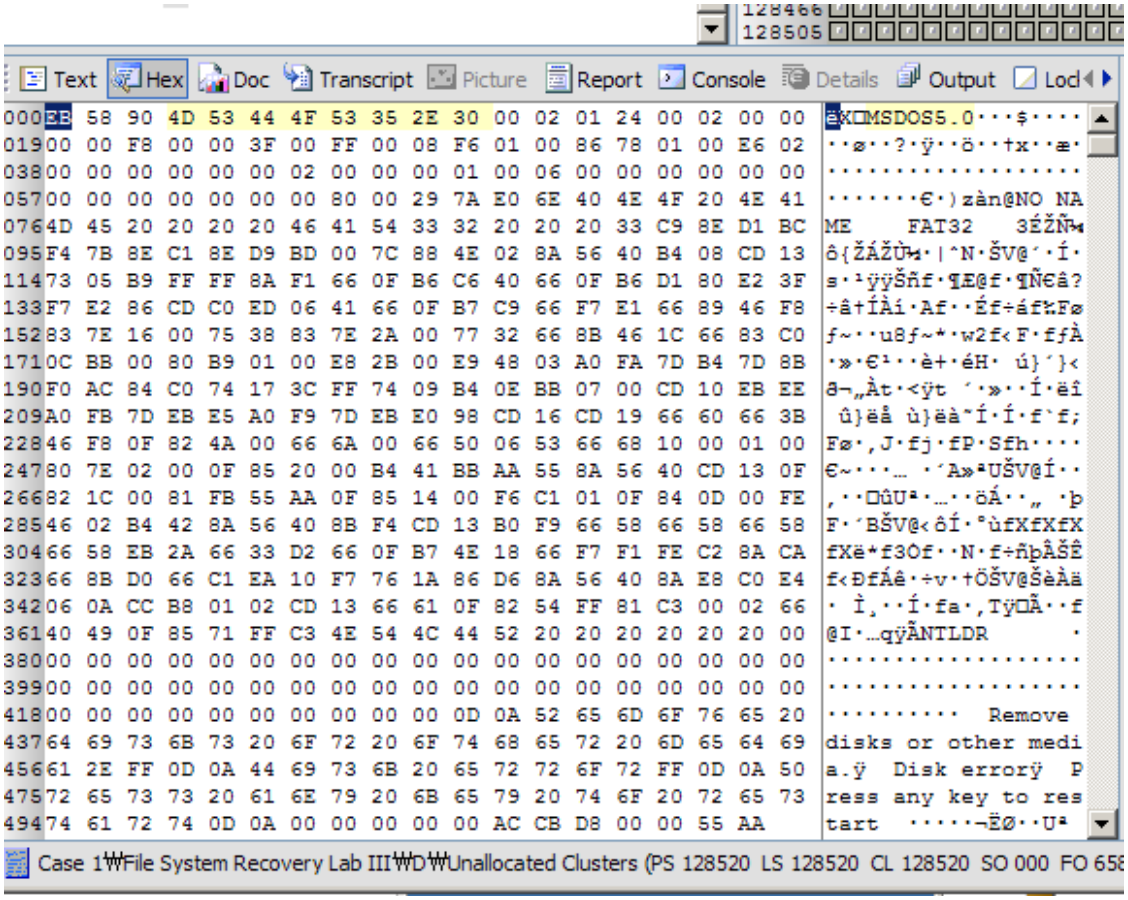


// 총 4 개가 검색되었다. 이때, VBR 의 백업본도 함께 검색되므로, 선별 작업이 필요하다. 보통 NTFS 의 경우 VBR 백업본은 가장 마지막 섹터이고, FAT32 의 경우 VBR 로부터 6 번째 섹터에 기록되므로 이 하드 디스크의 총 파티션 수는 2 개이며, NTFS 와, FAT32 파일 시스템을 사용하고, 각각 파티션 시작 위치(VBR)은 63 과 128520 임을 알 수 있다.

e. 두 번째 파티션(FAT32) 인식

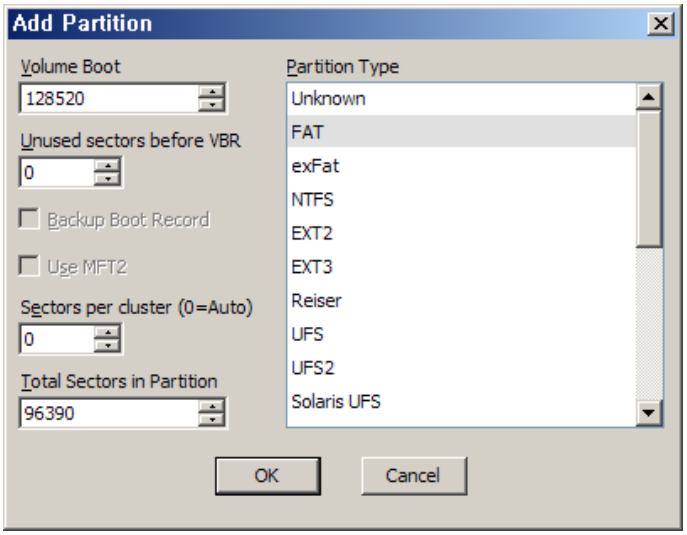
e-1. 시작 지점인 128520 섹터로 가 본다.

(첫 번째 파티션의 마지막 섹터 바로 다음이다.)



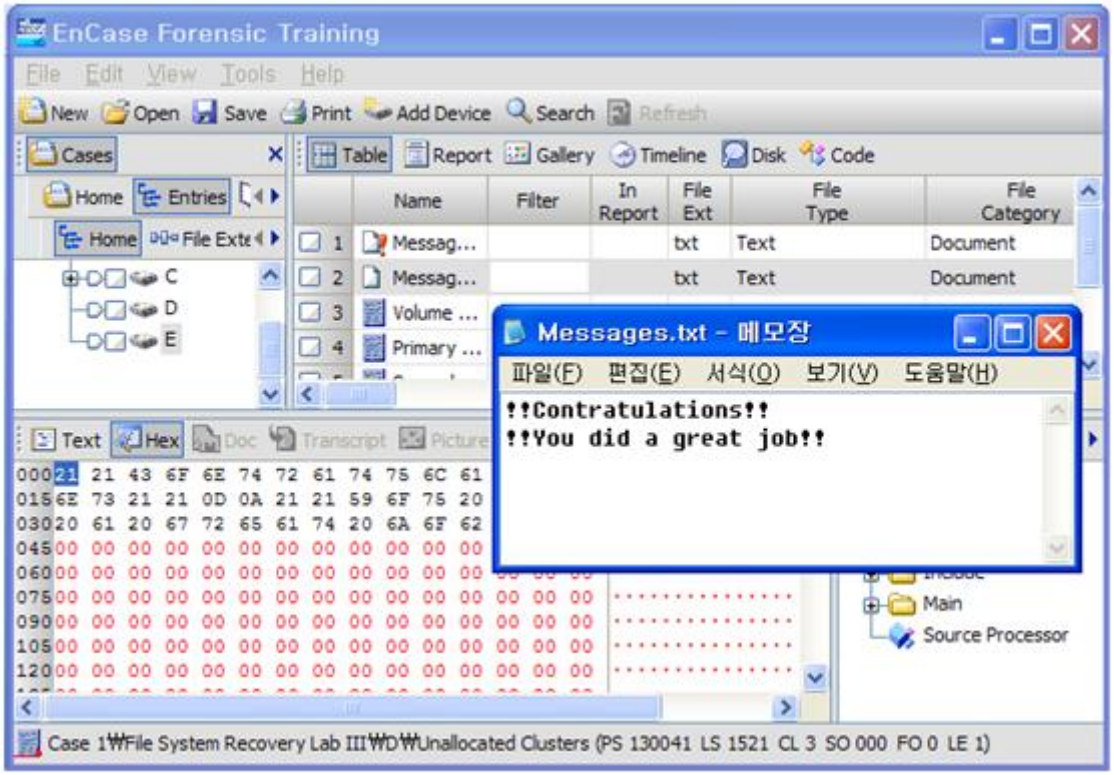
// 바로 VBR 이 위치하는 것을 알 수 있다. 첫 번째 파티션이 128519 에서 끝났으므로 두 번째 파티션 또한 Primary Partition 인 것을 알 수 있다. (BR 이 없으므로, MBR 에 등록되어있는 Primary 임을 의미함)

e-2. Add Partition



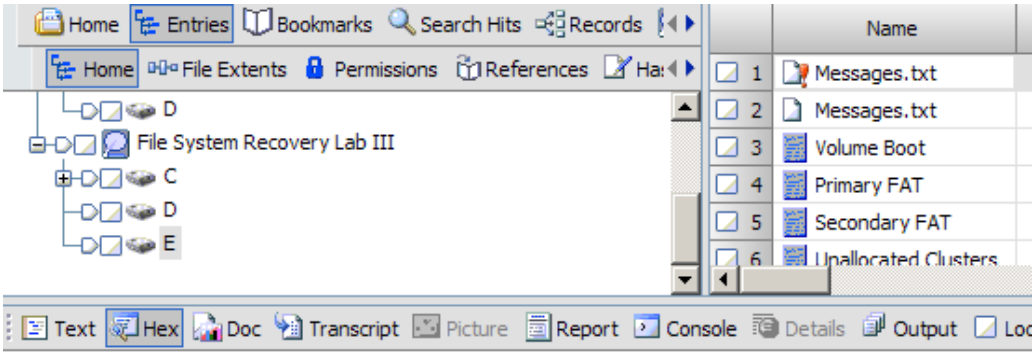
- **파티션 복원 완료**

  새로 추가된 E 드라이브의 Messages.txt 파일 내용 확인

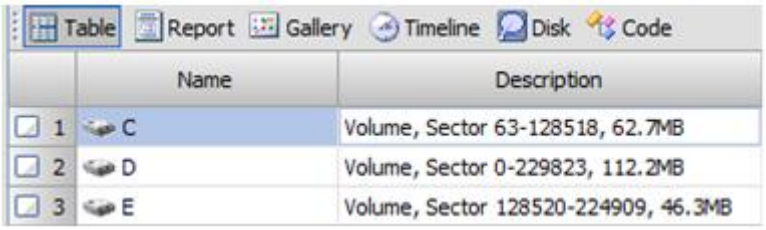


- **추가 연구가 필요한 사항**

  추가하려는 E 드라이브 외에 D 드라이브(C 와 E 전부를 가리킴)가 생성되는 이유와 이를 포렌식적으로 제거하는 방법에 대한 연구가 필요하다.



c.f) 최종 복구 후 모습



④ 각각의 Partition의 VBR이 모두 손상된 경우

- **복원 전 MBR : Partition Table Layout**

Bootable Flag	Starting CHS	Partition Type	Ending CHS	Starting LBA	Size in sector
00	01 01 00	07	FE 3F 07	3F 00 00 00	C9 F5 01 00
00	00 01 08	05	FE 3F 0D	08 F6 01 00	86 78 01 00

- **하드 전체 Layout**

MBR		Primary 1	Primary 2

- **하드디스크 정보 (229,824 Sectors, 112.2MB)**

Type	Name	Status	Start	Stop	Relative	size
07	NTFS	00	000101	073FFE	63	128457
05	FAT32	00	080100	0D3FFE	128520	96390

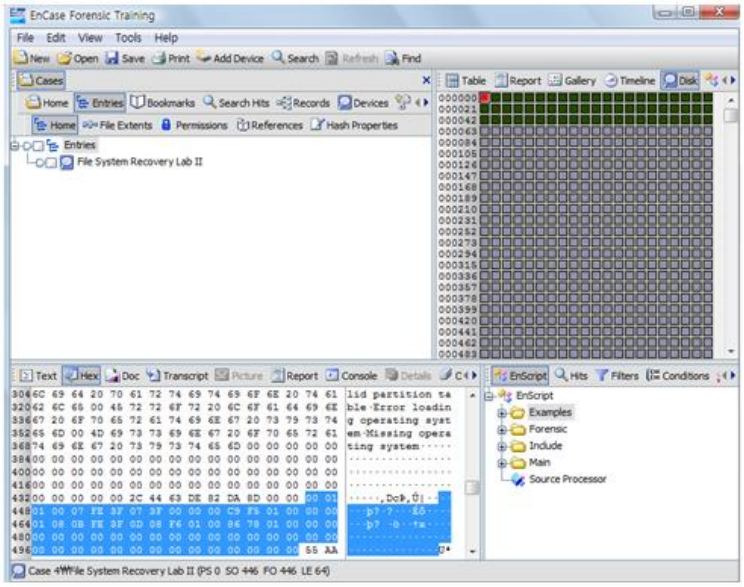
- **인식 오류 발생 원인**

Encase 로 불러왔을 때, 아무런 파티션도 인식하지 못한 상태이다.

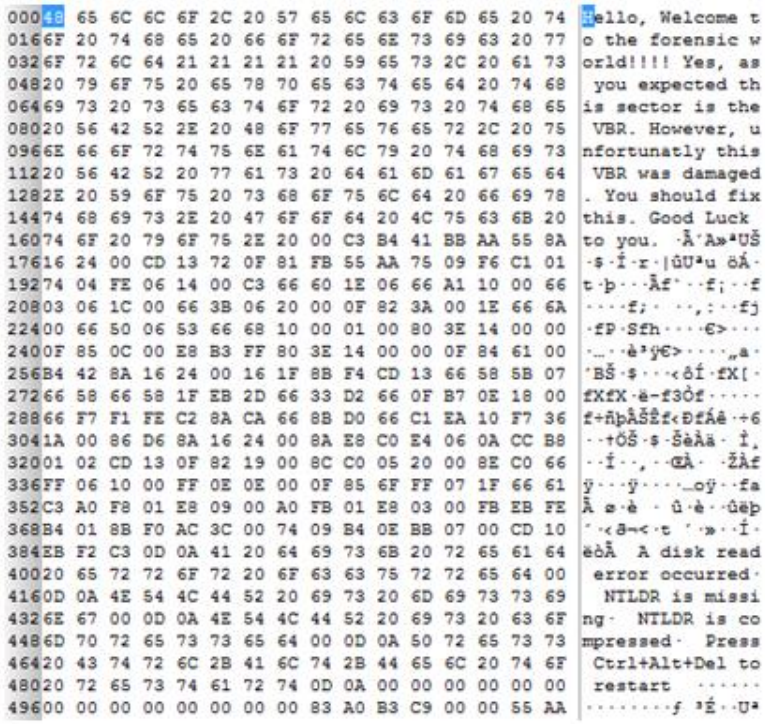
해당 디스크의 MBR 을 확인해 보았다. MBR 에는 2 개의 파티션 정보가 나타나 있다. 해당 정보에 따르면, 첫 번째 파티션 시작 위치는 63 섹터, 두 번째 파티션 시작 위치는 128520 섹터이다. 그러나 해당 섹터(VBR)는 모두 손상되었다.

즉, MBR 은 손상되지 않았지만, 각각 파티션들의 VBR 이 손상되었다.

a. MBR 의 Partition table 을 보고, 각 Partition 의 시작 위치 확인

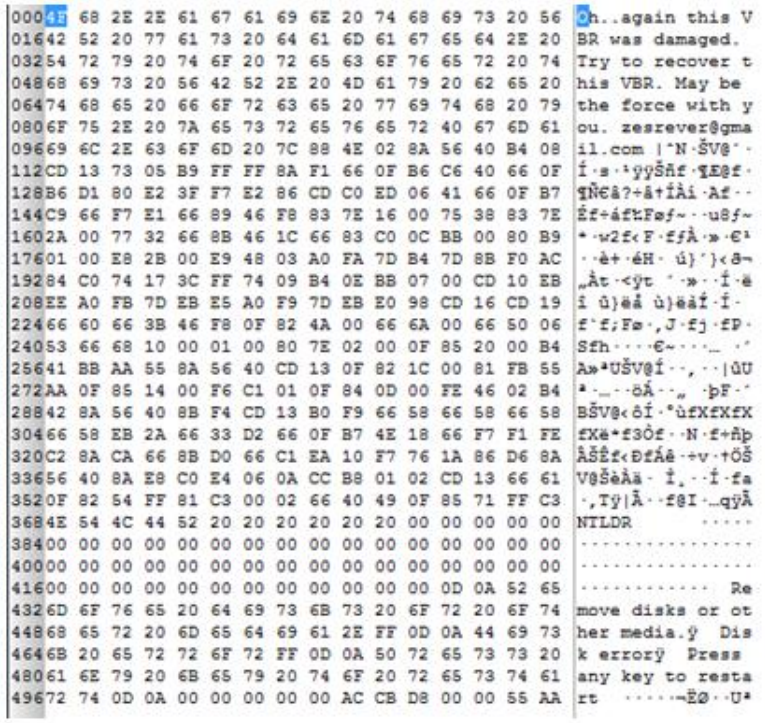


b. 첫 번째 파티션의 시작 위치인 63 sector 에서 VBR 손상 확인



```
000 48 65 6C 6C 6F 2C 20 57 65 6C 63 6F 6D 65 20 74 Hello, Welcome t
016 6F 20 74 68 65 20 66 6F 72 65 6E 73 69 63 20 77 o the forensic w
032 6F 72 6C 64 21 21 21 21 20 59 65 73 2C 20 61 73 orld!!!! Yes, as
048 20 79 6F 75 20 65 78 70 65 63 74 65 64 20 74 68  you expected th
064 69 73 20 73 65 63 74 6F 72 20 69 73 20 74 68 65 is sector is the
080 20 56 42 52 2E 20 48 6F 77 65 76 65 72 2C 20 75  VBR. However, u
096 6E 66 6F 72 74 75 6E 61 74 6C 79 20 74 68 69 73 nfortunatly this
112 20 56 42 52 20 77 61 73 20 64 61 6D 61 67 65 64  VBR was damaged
128 2E 20 59 6F 75 20 73 68 6F 75 6C 64 20 66 69 78 . You should fix
144 74 68 69 73 2E 20 47 6F 6F 64 20 4C 75 63 6B 20 this. Good Luck
160 74 6F 20 79 6F 75 2E 20 00 C3 B4 41 BB AA 55 8A to you. ·Ã´A»ªUŠ
176 16 24 00 CD 13 72 0F 81 FB 55 AA 75 09 F6 C1 01 ·$·Í·r·ûUªu öÁ·
192 74 04 FE 06 14 00 C3 66 60 1E 06 66 A1 10 00 66 t·þ···Ãf`··f¡··f
208 03 06 1C 00 66 3B 06 20 00 0F 82 3A 00 1E 66 6A ····f;· ··,:··fj
224 00 66 50 06 53 66 68 10 00 01 00 80 3E 14 00 00 ·fP·Sfh····€>···
240 0F 85 0C 00 E8 B3 FF 80 3E 14 00 00 0F 84 61 00 ·…··è³ÿ€>····„a·
256 B4 42 8A 16 24 00 16 1F 8B F4 CD 13 66 58 5B 07 ´BŠ·$···‹ôÍ·fX[·
272 66 58 66 58 1F EB 2D 66 33 D2 66 0F B7 0E 18 00 fXfX·ë-f3Òf·····
288 66 F7 F1 FE C2 8A CA 66 8B D0 66 C1 EA 10 F7 36 f÷ñþÂŠÊf‹ÐfÁê·÷6
304 1A 00 86 D6 8A 16 24 00 8A E8 C0 E4 06 0A CC B8 ··†ÖŠ·$·ŠèÀä·Ì¸
320 01 02 CD 13 0F 82 19 00 8C C0 05 20 00 8E C0 66 ··Í···,··ŒÀ· ·ŽÀf
336 FF 06 10 00 FF 0E 0E 00 0F 85 6F FF 07 1F 66 61 ÿ···ÿ·····…oÿ··fa
352 C3 A0 F8 01 E8 09 00 A0 FB 01 E8 03 00 FB EB FE Ã ø·è· ·û·è··ûëþ
368 B4 01 8B F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 ´·‹ð¬<·t ´·»··Í·
384 EB F2 C3 0D 0A 41 20 64 69 73 6B 20 72 65 61 64 ëòÃ  A disk read
400 20 65 72 72 6F 72 20 6F 63 63 75 72 72 65 64 00  error occurred·
416 0D 0A 4E 54 4C 44 52 20 69 73 20 6D 69 73 73 69   NTLDR is missi
432 6E 67 00 0D 0A 4E 54 4C 44 52 20 69 73 20 63 6F ng·  NTLDR is co
448 6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 73 mpressed·  Press
464 20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 6F  Ctrl+Alt+Del to
480 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00 00  restart  ·····
496 00 00 00 00 00 00 00 00 83 A0 B3 C9 00 00 55 AA ·····ƒ ³É··Uª
```

c. 두 번째 파티션의 시작 위치인 128520 sector 에서 VBR 손상 확인



```
000 4E 68 2E 2E 61 67 61 69 6E 20 74 68 69 73 20 56 Nh..again this V
016 42 52 20 77 61 73 20 64 61 6D 61 67 65 64 2E 20 BR was damaged.
032 54 72 79 20 74 6F 20 72 65 63 6F 76 65 72 20 74 Try to recover t
048 68 69 73 20 56 42 52 2E 20 4D 61 79 20 62 65 20 his VBR. May be
064 74 68 65 20 66 6F 72 63 65 20 77 69 74 68 20 79 the force with y
080 6F 75 2E 20 7A 65 73 72 65 76 65 72 40 67 6D 61 ou. zesrever@gma
096 69 6C 2E 63 6F 6D 20 7C 88 4E 02 8A 56 40 B4 08 il.com |ˆN·ŠV@´·
112 CD 13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 66 0F Í·s·¹ÿÿŠñf·¶Æ@f·
128 B6 D1 80 E2 3F F7 E2 86 CD C0 ED 06 41 66 0F B7 ¶Ñ€â?÷â†ÍÀí·Af··
144 C9 66 F7 E1 66 89 46 F8 83 7E 16 00 75 38 83 7E Éf÷áf‰Føƒ~··u8ƒ~
160 2A 00 77 32 66 8B 46 1C 66 83 C0 0C BB 00 80 B9 *·w2f‹F·fƒÀ·»·€¹
176 01 00 E8 2B 00 E9 48 03 A0 FA 7D B4 7D 8B F0 AC ··è+·éH· ú}´}‹ð¬
192 84 C0 74 17 3C FF 74 09 B4 0E BB 07 00 CD 10 EB „Àt·<ÿt ´·»··Í·ë
208 EE A0 FB 7D EB E5 A0 F9 7D EB E0 98 CD 16 CD 19 î û}ëå ù}ëà˜Í·Í·
224 66 60 66 3B 46 F8 0F 82 4A 00 66 6A 00 66 50 06 f`f;Fø·‚J·fj·fP·
240 53 66 68 10 00 01 00 80 7E 02 00 0F 85 20 00 B4 Sfh····€~···… ·´
256 41 BB AA 55 8A 56 40 CD 13 0F 82 1C 00 81 FB 55 A»ªUŠV@Í··‚··|ûU
272 AA 0F 85 14 00 F6 C1 01 0F 84 0D 00 FE 46 02 B4 ª·…··öÁ··„ ·þF·´
288 42 8A 56 40 8B F4 CD 13 B0 F9 66 58 66 58 66 58 BŠV@‹ôÍ·°ùfXfXfX
304 66 58 EB 2A 66 33 D2 66 0F B7 4E 18 66 F7 F1 FE fXë*f3Òf·‚·N·f÷ñþ
320 C2 8A CA 66 8B D0 66 C1 EA 10 F7 76 1A 86 D6 8A ÂŠÊf‹ÐfÁê·÷v·†ÖŠ
336 56 40 8A E8 C0 E4 06 0A CC B8 01 02 CD 13 66 61 V@ŠèÀä· Ì¸··Í·fa
352 0F 82 54 FF 81 C3 00 02 66 40 49 0F 85 71 FF C3 ·‚Tÿ|Ã··f@I·…qÿÃ
368 4E 54 4C 44 52 20 20 20 20 20 20 00 00 00 00 00 NTLDR       ·····
384 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····------------
400 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····------------
416 00 00 00 00 00 00 00 00 00 00 00 00 0D 0A 52 65 ··------------ Re
432 6D 6F 76 65 20 64 69 73 6B 73 20 6F 72 20 6F 74 move disks or ot
448 68 65 72 20 6D 65 64 69 61 2E FF 0D 0A 44 69 73 her media.ÿ  Dis
464 6B 20 65 72 72 6F 72 FF 0D 0A 50 72 65 73 73 20 k errorÿ  Press
480 61 6E 79 20 6B 65 79 20 74 6F 20 72 65 73 74 61 any key to resta
496 72 74 0D 0A 00 00 00 00 00 AC CB D8 00 00 55 AA rt  ·····¬ËØ··Uª
```

- **복원 과정**

  VBR 이 손상되었을 때에는, VBR 백업본을 이용하여 복원할 수 있는 가능성이 존재한다. NTFS 의 경우 파티션의 가장 마지막 섹터에 백업 되며, FAT32 의 경우 보통 VBR 에서 6 번째 섹터에 기록되어 있다. 각각 파티션의 백업 VBR 의 존재를 확인 후 손상된 VBR 의 위치에 덮어쓰는 방법으로 복원할 수 있다.

  a. 첫 번째 파티션의 VBR (63 sector) 바로 다음 섹터 확인

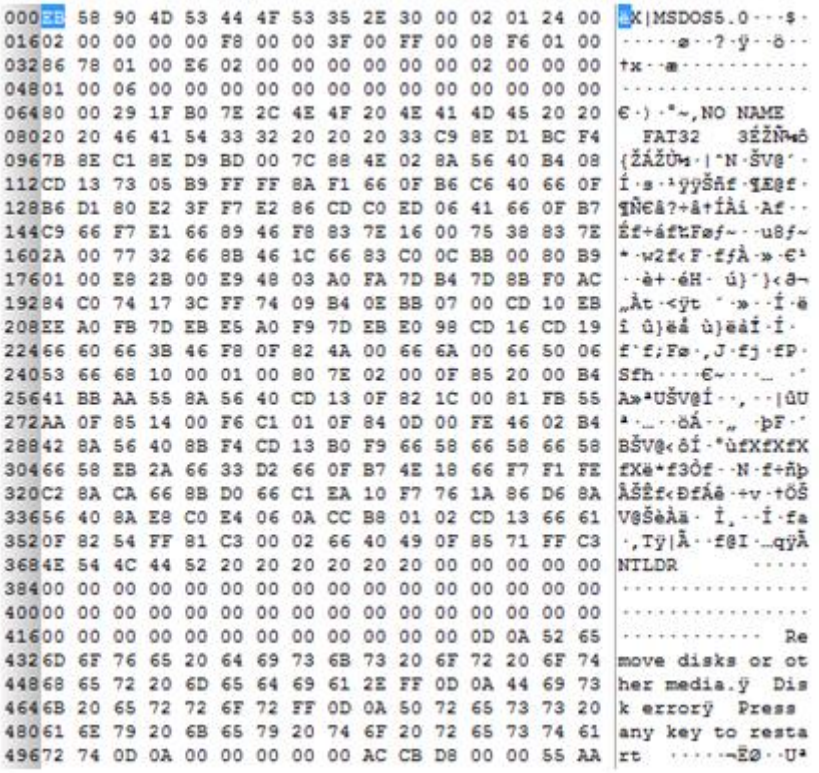
// NTLDR 관련 정보가 있는 것으로 보아 NTFS 인 것으로 추측.
NTFS 의 경우 VBR 의 백업본이 맨 마지막 sector 에 저장되어 있음

b. 첫 번째 파티션의 맨 마지막 섹터 (128519 sector) 확인

```
000 EB 52 90 4E 54 46 53 20 20 20 20 00 02 01 00 00  ëR|NTFS     ·····
016 00 00 00 00 00 F8 00 00 3F 00 FF 00 3F 00 00 00  ·····ø··?·ÿ·?···
032 00 00 00 00 80 00 80 00 C8 F5 01 00 00 00 00 00  ····€·€·Èõ·····
048 43 A7 00 00 00 00 00 00 E4 FA 00 00 00 00 00 00  C§·····äú·····
064 02 00 00 00 08 00 00 00 55 36 76 48 4F 76 48 8E  ·····U6vHOvHŽ
080 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB B8 C0 07  ····ú3ÀŽÐ¼·|û¸À·
096 8E D8 E8 16 00 B8 00 0D 8E C0 33 DB C6 06 0E 00  ŽØè··¸··ŽÀ3ÛÆ···
112 10 E8 53 00 68 00 0D 68 6A 02 CB 8A 16 24 00 B4  ·èS·h··hj·ËŠ·$·´
128 08 CD 13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 66  ·Í·s·¹ÿÿŠñf·¶Æ@f
144 0F B6 D1 80 E2 3F F7 E2 86 CD C0 ED 06 41 66 0F  ·¶Ñ€â?÷â†ÍÀí·Af·
160 B7 C9 66 F7 E1 66 A3 20 00 C3 B4 41 BB AA 55 8A  ·Éf÷áf£ ·Ã´A»ªUŠ
176 16 24 00 CD 13 72 0F 81 FB 55 AA 75 09 F6 C1 01  ·$·Í·r·|ûUªu·öÁ·
192 74 04 FE 06 14 00 C3 66 60 1E 06 66 A1 10 00 66  t·þ···Ãf`··f¡··f
208 03 06 1C 00 66 3B 06 20 00 0F 82 3A 00 1E 66 6A  ····f;· ··,:··fj
224 00 66 50 06 53 66 68 10 00 01 00 80 3E 14 00 00  ·fP·Sfh····€>···
240 0F 85 0C 00 E8 B3 FF 80 3E 14 00 00 0F 84 61 00  ·…··è³ÿ€>····„a·
256 B4 42 8A 16 24 00 16 1F 8B F4 CD 13 66 58 5B 07  ´BŠ·$···‹ôÍ·fX[·
272 66 58 66 58 1F EB 2D 66 33 D2 66 0F B7 0E 18 00  fXfX·ë-f3Òf·····
288 66 F7 F1 FE C2 8A CA 66 8B D0 66 C1 EA 10 F7 36  f÷ñþÂŠÊf‹ÐfÁê·÷6
304 1A 00 86 D6 8A 16 24 00 8A E8 C0 E4 06 0A CC B8  ··†ÖŠ·$·ŠèÀä··Ì¸
320 01 02 CD 13 0F 82 19 00 8C C0 05 20 00 8E C0 66  ··Í··‚··ŒÀ·· ·ŽÀf
336 FF 06 10 00 FF 0E 0E 00 0F 85 6F FF 07 1F 66 61  ÿ···ÿ·····…oÿ··fa
352 C3 A0 F8 01 E8 09 00 A0 FB 01 E8 03 00 FB EB FE  Ã ø·è·· ·û·è···ûëþ
368 B4 01 8B F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10  ´·‹ð¬<·t·´·»··Í·
384 EB F2 C3 0D 0A 41 20 64 69 73 6B 20 72 65 61 64  ëòÃ··A disk read
400 20 65 72 72 6F 72 20 6F 63 63 75 72 72 65 64 00   error occurred·
416 0D 0A 4E 54 4C 44 52 20 69 73 20 6D 69 73 73 69  ··NTLDR is missi
432 6E 67 00 0D 0A 4E 54 4C 44 52 20 69 73 20 63 6F  ng···NTLDR is co
448 6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 73  mpressed···Press
464 20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 6F   Ctrl+Alt+Del to
480 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00 00   restart ·····
496 00 00 00 00 00 00 00 00 83 A0 B3 C9 00 00 55 AA  ·····ƒ ³É··Uª
```

// NTFS 파일시스템 파티션 VBR 이 백업되어 있음

c. 두 번째 파티션의 시작위치(128520 sector)로부터 6 sector 뒤 확인

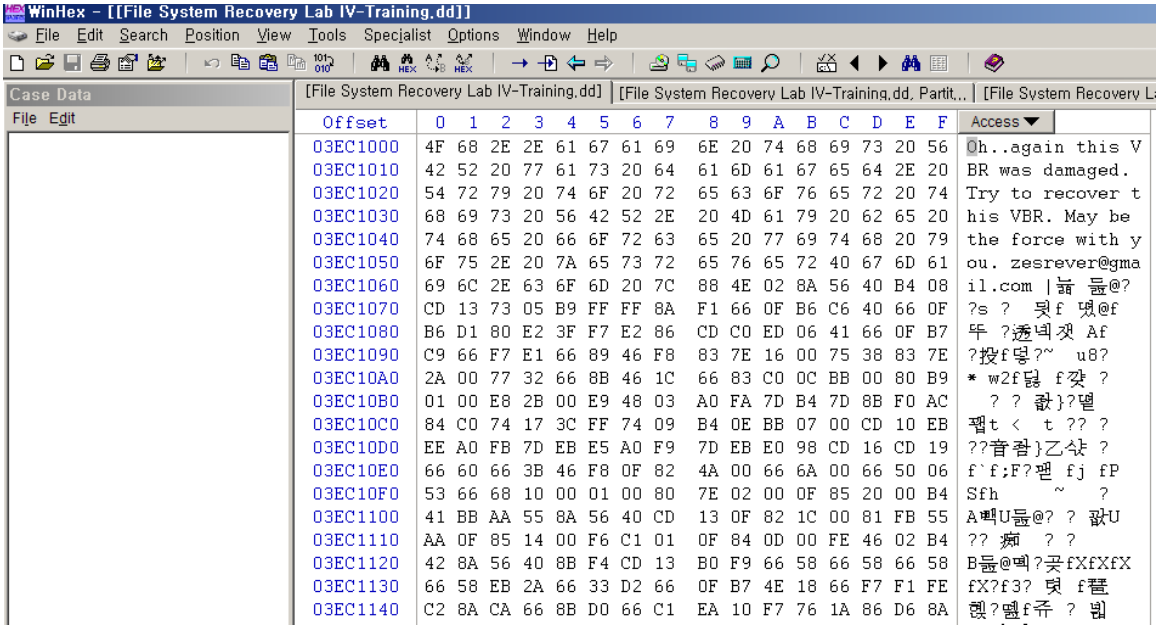


```
000 EB 58 90 4D 53 44 4F 53 35 2E 30 00 02 01 24 00  ëX|MSDOS5.0···$·
016 02 00 00 00 00 F8 00 00 3F 00 FF 00 08 F6 01 00  ·····ø··?·ÿ··ö··
032 86 78 01 00 E6 02 00 00 00 00 00 00 02 00 00 00  †x··æ···········
048 01 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00  ···············
064 80 00 29 1F B0 7E 2C 4E 4F 20 4E 41 4D 45 20 20  €·)·°~,NO NAME
080 20 20 46 41 54 33 32 20 20 20 33 C9 8E D1 BC F4    FAT32   3ÉŽÑ¼ô
096 7B 8E C1 8E D9 BD 00 7C 88 4E 02 8A 56 40 B4 08  {ŽÁŽÙ½·|ˆN·ŠV@´·
112 CD 13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 66 0F  Í·s·¹ÿÿŠñf·¶Æ@f·
128 B6 D1 80 E2 3F F7 E2 86 CD C0 ED 06 41 66 0F B7  ¶Ñ€â?÷â†ÍÀí·Af··
144 C9 66 F7 E1 66 89 46 F8 83 7E 16 00 75 38 83 7E  Éf÷áf‰Føƒ~··u8ƒ~
160 2A 00 77 32 66 8B 46 1C 66 83 C0 0C BB 00 80 B9  *·w2f‹F·fƒÀ·»·€¹
176 01 00 E8 2B 00 E9 48 03 A0 FA 7D B4 7D 8B F0 AC  ··è+·éH· ú}´}‹ð¬
192 84 C0 74 17 3C FF 74 09 B4 0E BB 07 00 CD 10 EB  „Àt·<ÿt·´·»··Í·ë
208 EE A0 FB 7D EB E5 A0 F9 7D EB E0 98 CD 16 CD 19  î û}ëå ù}ëà˜Í·Í·
224 66 60 66 3B 46 F8 0F 82 4A 00 66 6A 00 66 50 06  f`f;Fø·‚J·fj·fP·
240 53 66 68 10 00 01 00 80 7E 02 00 0F 85 20 00 B4  Sfh····€~···… ·´
256 41 BB AA 55 8A 56 40 CD 13 0F 82 1C 00 81 FB 55  A»ªUŠV@Í··‚··|ûU
272 AA 0F 85 14 00 F6 C1 01 0F 84 0D 00 FE 46 02 B4  ª·…··öÁ··„··þF·´
288 42 8A 56 40 8B F4 CD 13 B0 F9 66 58 66 58 66 58  BŠV@‹ôÍ·°ùfXfXfX
304 66 58 EB 2A 66 33 D2 66 0F B7 4E 18 66 F7 F1 FE  fXë*f3Òf·N·f÷ñþ
320 C2 8A CA 66 8B D0 66 C1 EA 10 F7 76 1A 86 D6 8A  ÂŠÊf‹ÐfÁê·÷v·†ÖŠ
336 56 40 8A E8 C0 E4 06 0A CC B8 01 02 CD 13 66 61  V@ŠèÀä··Ì¸··Í·fa
352 0F 82 54 FF 81 C3 00 02 66 40 49 0F 85 71 FF C3  ·‚Tÿ|Ã··f@I·…qÿÃ
368 4E 54 4C 44 52 20 20 20 20 20 20 00 00 00 00 00  NTLDR      ·····
384 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ················
400 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ················
416 00 00 00 00 00 00 00 00 00 00 00 00 0D 0A 52 65  ··············Re
432 6D 6F 76 65 20 64 69 73 6B 73 20 6F 72 20 6F 74  move disks or ot
448 68 65 72 20 6D 65 64 69 61 2E FF 0D 0A 44 69 73  her media.ÿ··Dis
464 6B 20 65 72 72 6F 72 FF 0D 0A 50 72 65 73 73 20  k errorÿ··Press
480 61 6E 79 20 6B 65 79 20 74 6F 20 72 65 73 74 61  any key to resta
496 72 74 0D 0A 00 00 00 00 00 AC CB D8 00 00 55 AA  rt·····¬ËØ··Uª
```

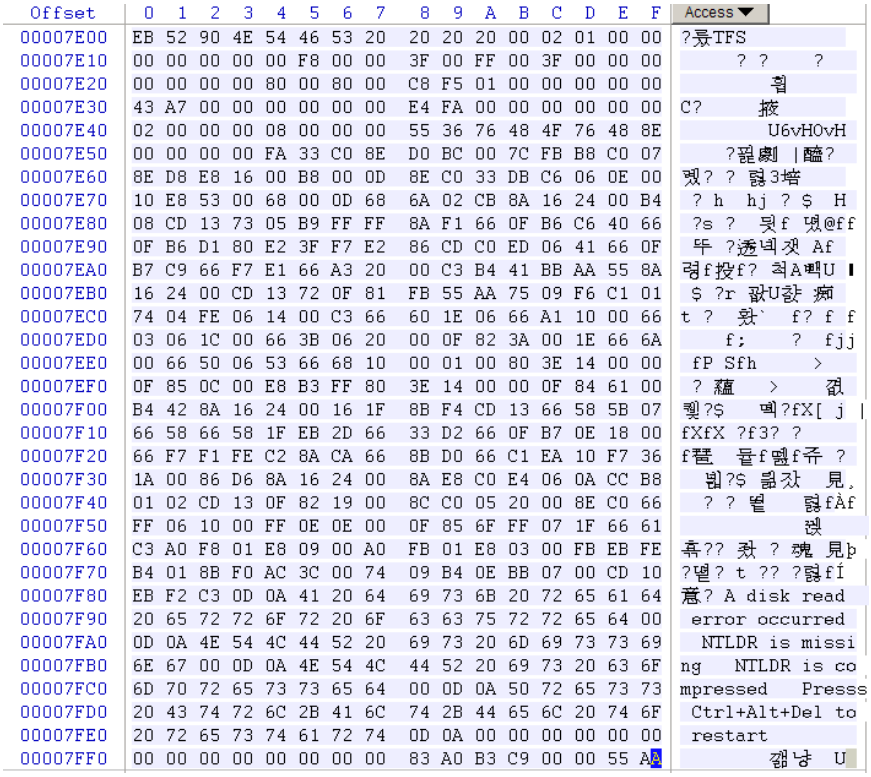
// FAT32 파일시스템 파티션 VBR 이 백업되어 있음

d. WinHex 로 dd 이미지를 읽음

(Encase Forensic Training 버전에서는 Disk Emulate 기능 없으므로 직접 수정이 불가하므로 WinHex 를 이용하여 VBR 백업본을 직접 손상된 VBR 에 덮어써서 복원 후 마운트해야 함)



e. Encase 를 통해 이미 확인한 VBR 백업본을 원래의 VBR 위치에 덮어씌움

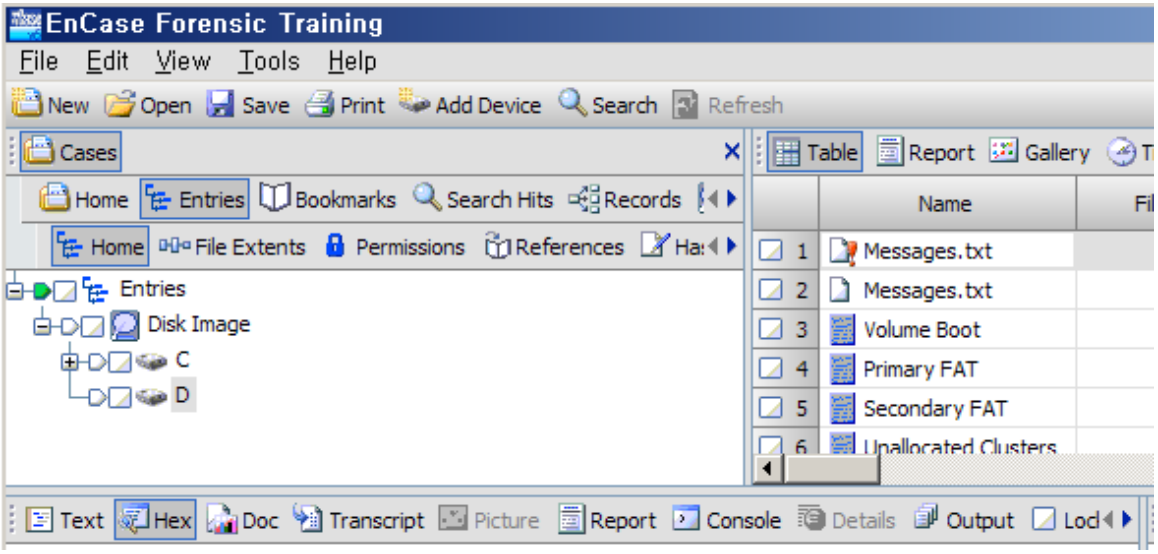


// NTFS 의 VBR 백업본

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Access ▼
03EC1000	EB	58	90	4D	53	44	4F	53	35	2E	30	00	02	01	24	00	ëX♥MSDOS5.0 $
03EC1010	02	00	00	00	00	F8	00	00	3F	00	FF	00	08	F6	01	00	ø ? ÿ ö
03EC1020	86	78	01	00	E6	02	00	00	00	00	00	00	02	00	00	00	♥x æ
03EC1030	01	00	06	00	00	00	00	00	00	00	00	00	00	00	00	00	
03EC1040	80	00	29	1F	B0	7E	2C	4E	4F	20	4E	41	4D	45	20	20	€) °~,NO NAME
03EC1050	20	20	46	41	54	33	32	20	20	20	33	C9	8E	D1	BC	F4	FAT32 3É♫Ñ¼ô
03EC1060	7B	8E	C1	8E	D9	BD	00	7C	88	4E	02	8A	56	40	B4	08	{♫Á♫Ù½ |♦N ◘V@´
03EC1070	CD	13	73	05	B9	FF	FF	8A	F1	66	0F	B6	C6	40	66	0F	Í ↕s ¹ÿÿ♫ñf ¶Æ@f
03EC1080	B6	D1	80	E2	3F	F7	E2	86	CD	C0	ED	06	41	66	0F	B7	¶Ñ♫â?÷â♫ÍÀí♠Af ··
03EC1090	C9	66	F7	E1	66	89	46	F8	83	7E	16	00	75	38	83	7E	Éf÷áf◘Fø♫~ u8♫~
03EC10A0	2A	00	77	32	66	8B	46	1C	66	83	C0	0C	BB	00	80	B9	* w2f♫F f♫À » ♫¹
03EC10B0	01	00	E8	2B	00	E9	48	03	A0	FA	7D	B4	7D	8B	F0	AC	☺ è+ éH ♠ú}´}♫ð¬
03EC10C0	84	C0	74	17	3C	FF	74	09	B4	0E	BB	07	00	CD	10	EB	ä♫At <ÿt ♣´ ♫» ♫Í ♫ë
03EC10D0	EE	A0	FB	7D	EB	E5	A0	F9	7D	EB	E0	98	CD	16	CD	19	î ♠û}ëå ♠ù}ëà♫Í ♫Í
03EC10E0	66	60	66	3B	46	F8	0F	82	4A	00	66	6A	00	66	50	06	f`f;Fø ☼‚J f♠j f♠P
03EC10F0	53	66	68	10	00	01	00	80	7E	02	00	0F	85	20	00	B4	Sfh ►☺ ♫~ ☻ ☼… ♠´
03EC1100	41	BB	AA	55	8A	56	40	CD	13	0F	82	1C	00	81	FB	55	A»ªU♫V@Í ↕ ♫‚∟ ♠ü♠U
03EC1110	AA	0F	85	14	00	F6	C1	01	0F	84	0D	00	FE	46	02	B4	ª☼… ¶ ♠öÁ ☺ ♫„♪ þF ☻´
03EC1120	42	8A	56	40	8B	F4	CD	13	B0	F9	66	58	66	58	66	58	B♫V@♫ôÍ ↕°ùfXfXfX
03EC1130	66	58	EB	2A	66	33	D2	66	0F	B7	4E	18	66	F7	F1	FE	fXë*f3Òf ·N f÷ñþ
03EC1140	C2	8A	CA	66	8B	D0	66	C1	EA	10	F7	76	1A	86	D6	8A	Â♫Êf♫ÐfÁê ►÷v →Ö♫
03EC1150	56	40	8A	E8	C0	E4	06	0A	CC	B8	01	02	CD	13	66	61	V@♫èÀä ♠◙ Ì¸ ☻☺Í ‼fa
03EC1160	0F	82	54	FF	81	C3	00	02	66	40	49	0F	85	71	FF	C3	☼‚Tÿ♫Ã ☻♠f@I ☼…qÿÃ
03EC1170	4E	54	4C	44	52	20	20	20	20	20	20	00	00	00	00	00	NTLDR
03EC1180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
03EC1190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
03EC11A0	00	00	00	00	00	00	00	00	00	00	00	00	0D	0A	52	65	Re
03EC11B0	6D	6F	76	65	20	64	69	73	6B	73	20	6F	72	20	6F	74	move disks or ot
03EC11C0	68	65	72	20	6D	65	64	69	61	2E	FF	0D	0A	44	69	73	her media.ÿ Dis
03EC11D0	6B	20	65	72	72	6F	72	FF	0D	0A	50	72	65	73	73	20	k errorÿ Press
03EC11E0	61	6E	79	20	6B	65	79	20	74	6F	20	72	65	73	74	61	any key to resta
03EC11F0	72	74	0D	0A	00	00	00	00	00	AC	CB	D8	00	00	55	AA	rt ¬Ë0 Uª

// FAT32 의 VBR 백업본

- **복원 완료**
  a. Encase 로 확인
     Winhex 에서 저장 후 Encase 에서 Raw Image 로 불러들여 읽음



// 정상적으로 C, D 드라이브가 인식된 것을 확인

b. Linux 의 Loop Device 로 Mount 하여 정상 인식되는지 확인

b-1. dd 파일의 정보 확인 (Encase 로 확인한 바와 동일함)

```
bt home # sfdisk -l -uS filesystemrecovery.dd
Disk filesystemrecovery.dd: cannot get geometry

Disk filesystemrecovery.dd: 0 cylinders, 0 heads, 0 sectors/track
Warning: The partition table looks like it was made
  for C/H/S=*/255/63 (instead of 0/0/0).
For this listing I'll assume that geometry.
Units = sectors of 512 bytes, counting from 0

   Device Boot    Start       End   #sectors  Id  System
filesystemrecovery.dd1            63    128519     128457   7  HPFS/NTFS
filesystemrecovery.dd2        128520    224909      96390   b  W95 FAT32
filesystemrecovery.dd3             0         -          0   0  Empty
filesystemrecovery.dd4             0         -          0   0  Empty
```

c.f) MMLS filesystemrecovery.dd 를 이용하여도 Partition 정보 확인 가능

MMLS 는 Sleuthkit Tool 중 하나로, 분석할 디스크 레이아웃과 전체 파티션 스키마 취득, 물리적 디스크만 분석 가능 (스냅샵으로 획득한 이미지 파일에서는 사용 못함)

img_stat 도 스냅샷 작성한 파일과 원본 디스크 또는 파티션 정보 확인 가능

b-2. 첫 번째 파티션의 offset(시작위치 sector 를 byte 로 환산)을 지정하여 Mount 시킴

```
bt home # ls -al
total 115044
drwxr-xr-x  3 root root      4096 Jun  5 11:57 ./
drwxr-xr-x 20 root root      4096 Jun 18  2008 ../
-rwxrwxrwx  1 root root 117669888 Jun  5 12:53 filesystemrecovery.dd*
drwxr-xr-x  2 root root      4096 Jun  5 11:57 recovery/
bt home # sudo mount -o loop,offset=32256 -t auto /home/filesystemrecovery.dd /home/recovery
```

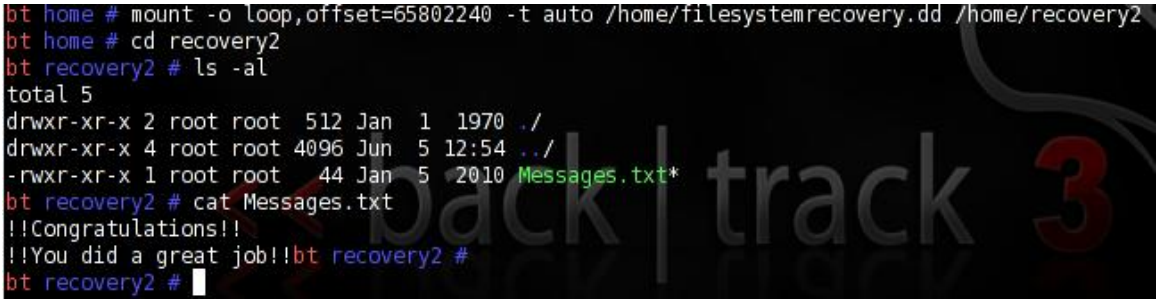
c.f) 파일 시스템을 마운트하는 것이므로, offset 을 지정하지 않는 경우 해당 이미지는 하나의 파일 내에 두 개의 파일시스템이 존재하게 되어 마운트 자체가 불가능함

"dd if=filesystemrecovery.dd of=recovery.dd bs=512 start=63 size=128457"와 같이 offset 을 지정하지 않고, 첫 번째 파티션만 따로 dump 하여 mount 가능

b-3. 마운트된 첫 번째 파티션 확인 (정상 인식)

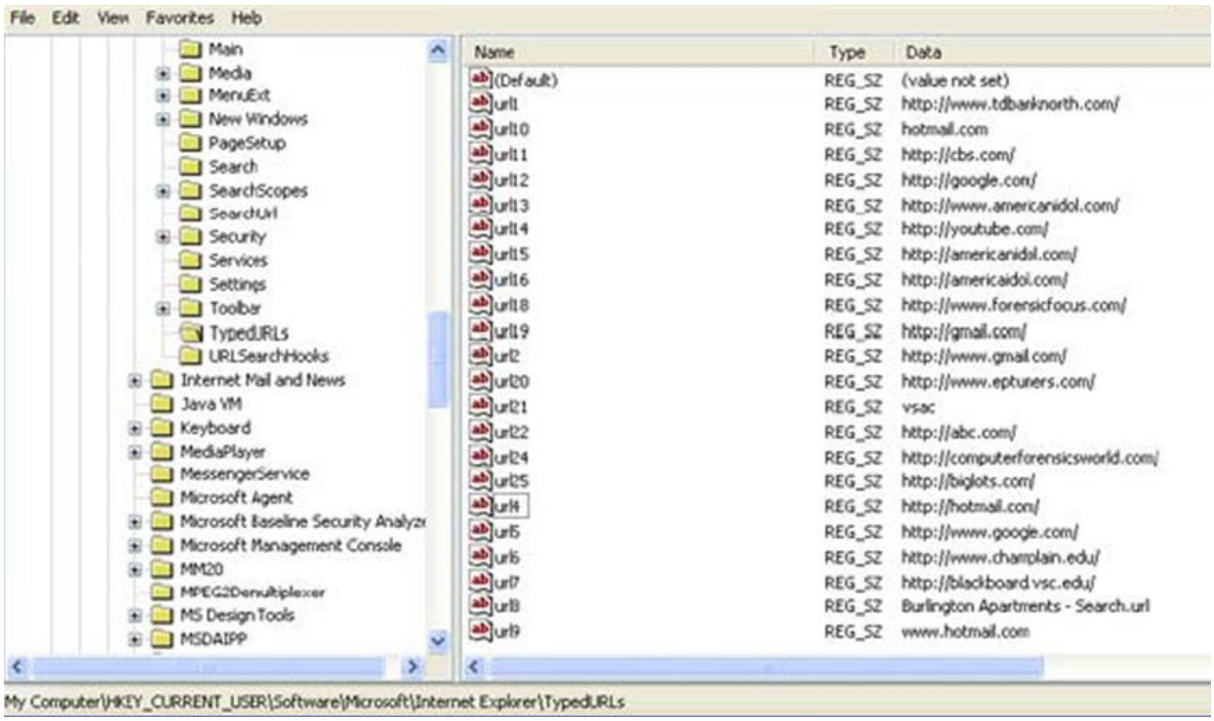
```
bt home # cd recovery
bt recovery # ls
Messages.txt  System\ Volume\ Information/
```

b-4. 두 번째 파티션 Mount 후 확인 (정상 인식)



7. Web History Analysis

Web History 라는 것은 피의자가 어떤 웹사이트에 언제 접속했는가를 찾기 위한 중요한 단서가 된다. (웹히스토리는 사용자가 브라우저를 통해 웹을 방문할 때, 그 방문 접속 기록을 가지고 있는 파일들이다.) 그리고 실제 수사에서도 가장 많이 사용되는 기록 중에 하나이다.

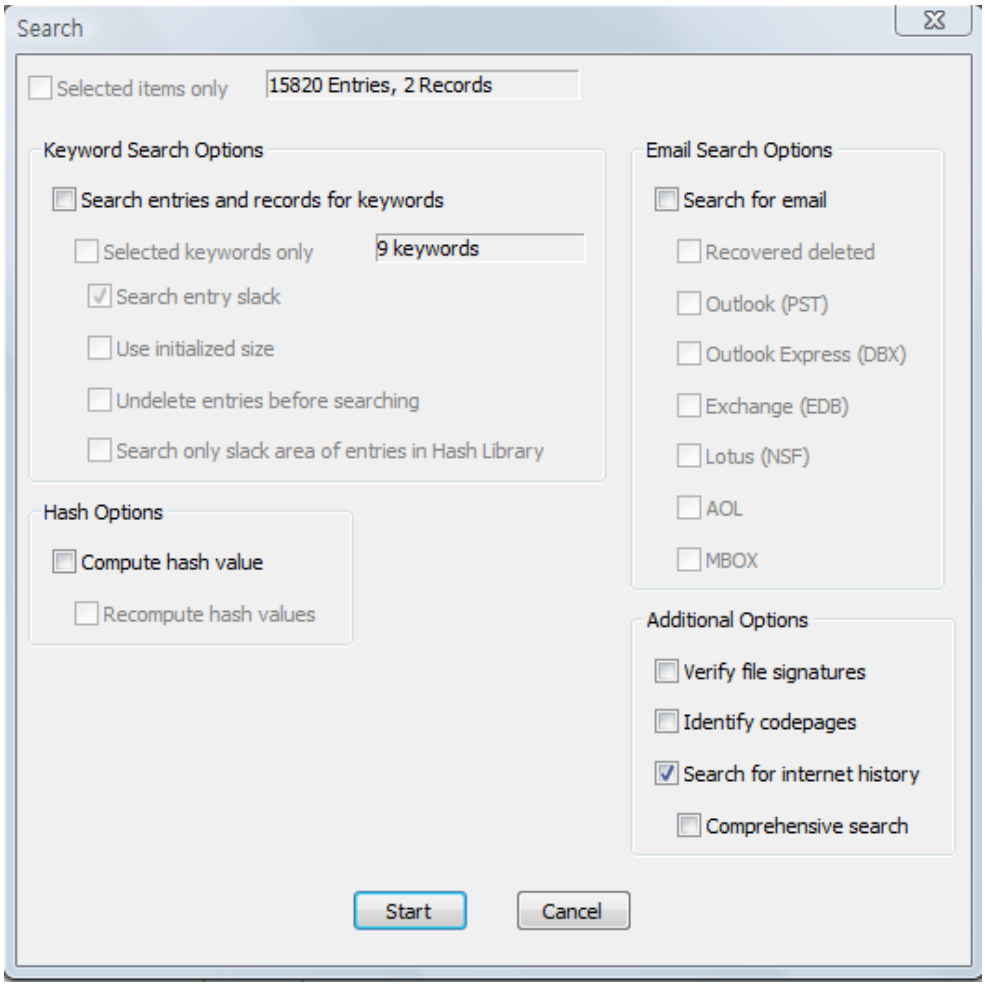


예를 들어 특정 사용자의 웹 히스토리 중 위와 같은 폴더를 확인했을 때, 이 데이터로부터 조사관은 사용자가 gmail과 hotmail 이메일 주소를 가지고 있고, tdbanknorth라는 온라인 뱅킹을 이용하며, 디지털 포렌식 웹사이트에 관심이 있고, 아마도 Champlain에 있는 대학을 다니고, 그 지역 아파트에 대해 알아보고 있다는 추측이 가능하다.[iii]

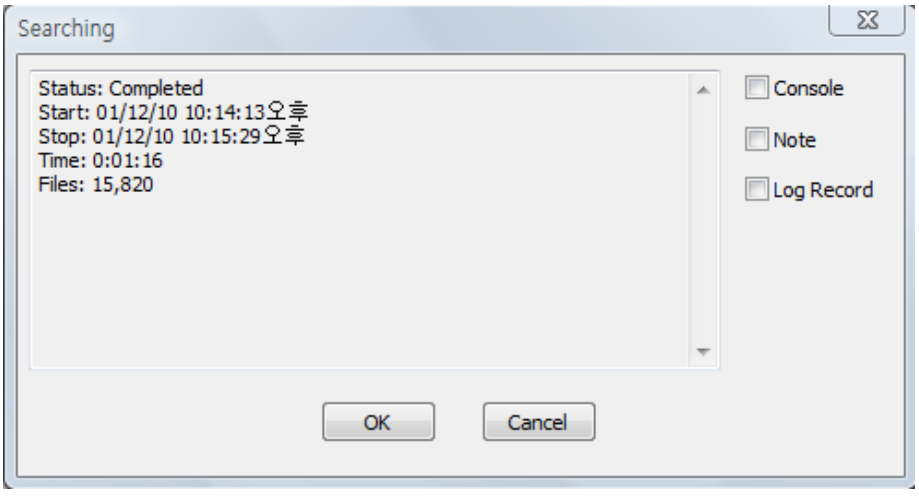
또한 다운로드 받은 파일을 마지막으로 저장한 폴더를 보여주는 경우, 조사관은 사용자가 파일들을 어디에 저장하는지 알 수 있으며, 이 밖에도 로그로 활용하기 위한 정보가 담겨있기도 하며, 이에는 웹 브라우져의 자동 로그온 아이디와 비밀번호, 검색어 등이 날짜, 시각 정보와 함께 저장되어 있다. 자동 완성 비밀번호 웹 페이지가 인코딩되어 저장되기도 한다.

Encase 에서 웹 히스토리를 분석하는 것은 간단하다.

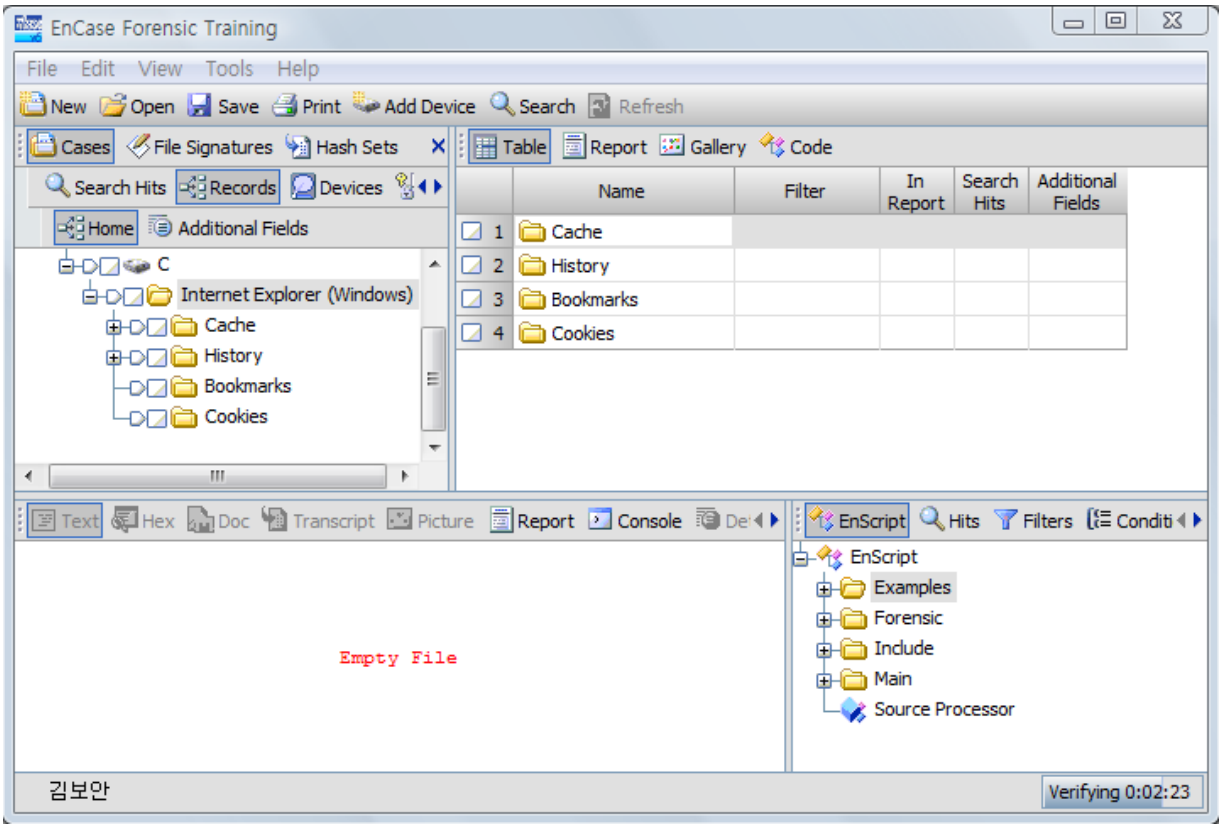
툴바의 Search 를 선택한다.



이 때 오른쪽 아래에 Search for internet history 를 체크해주면, 현재 추가된 증거들에서 자동으로 웹 히스토리 파일들을 찾아서 분류해준다.



결과는 Entries 탭이 아닌 Record 탭(View->Cases->Records)에서 볼 수 있다.



Encase 에서는 다음과 같은 웹히스토리 파일이 분석 가능하다.

- Internet Explorer - Windows, Mac
- Mozilla(Netscape, Firefox) - Windows, Mac
- Opera - Windows, Mac and Linux
- Safari - Mac

참고사항 : **Index.dat**

Internet Explorer의 경우 Index.dat 파일을 이용하여 브라우져의 사용에 관한 정보를 유지하는데, Index.dat 파일은 Cookies, History, Temporary Internet Files 폴더 등의 정보가 위치한다. 다양한 Index.dat 파일은 그 구조는 유사하지만, 서로 다른 타입의 데이터를 저장한다.

Cookies – 웹사이트에 의해 컴퓨터 시스템에 저장된 데이터.
.TXT 파일들이 실제 쿠키들을 나타내며,
Index.dat 파일은 각 쿠키 기록을 따른다.

Temporary Internet Files Index.dat – Internet Explorer를 통해 접근한 웹페이지에 나타난 파일들을 포함한다. 캐쉬 파일들은 컴퓨터에 저장될 때, 이름이 변경되는데 Index.dat는 이러한 변경 정보를 기록한다.

Main History Index.dat – 전반적인 현재 히스토리

Daily History Index.dat – 날짜 범위를 포함하는 폴더 저장.
예를 들어, 'MSHist012010050420100505'는 2010년 5월 4일 하루의 히스토리를 의미한다.

Weekly History Index.dat – 날짜 범위를 포함하는 폴더 저장.
예를 들어, 'MSHist012010050220100508'의 경우 2010년 5월 2일~8일 주를 의미한다.

위의 Main, Weekly, Daily History Index.dat는 Internet Explorer 외에도 Windows Explorer를 통해 접근한 Local 파일에 대한 히스토리도 기록한다.

8. Signature Analysis

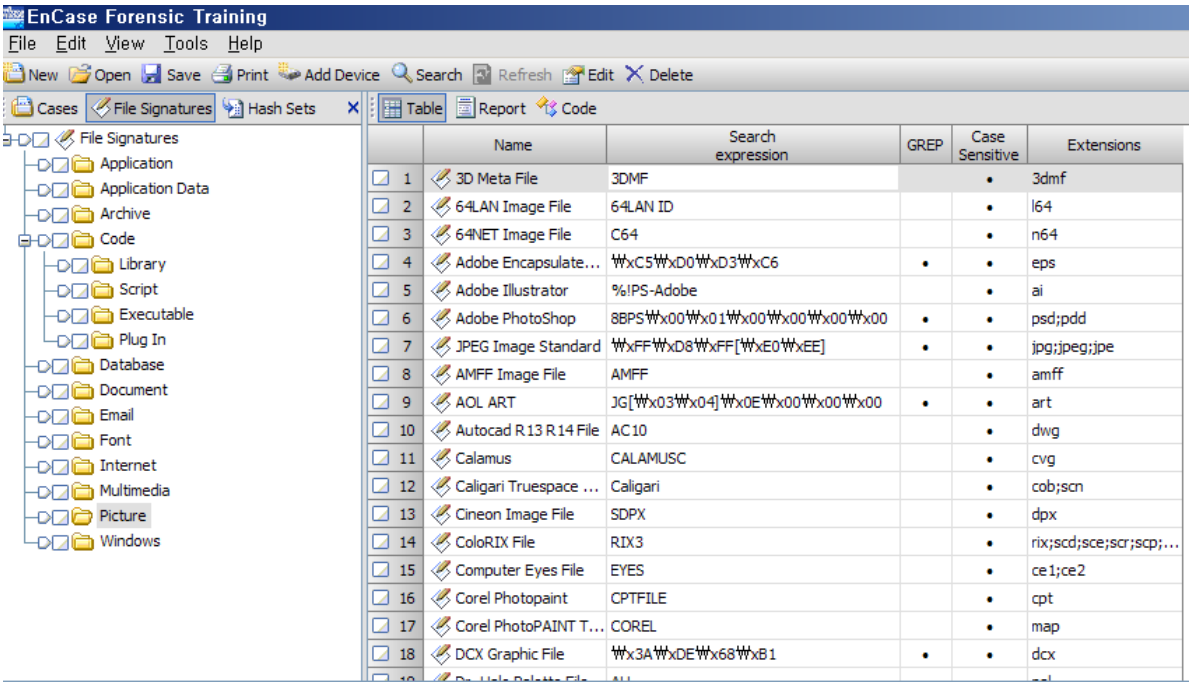
8-1. 개요

Signature Analysis는 확장자가 변경된 파일 식별이 주 목적이다.
대부분의 프로그램에서 확장자를 보고 파일 타입을 결정하는 것이 문제의 소지가 될 수 있으므로, 기록된 확장자와 파일의 실제 Signature 를 분석하여 일치하는 지를 확인하는 작업이다.

8-2. Signature

어떠한 파일타입인지를 나타내는 코드
보통 파일의 앞 부분
JPG 의 경우 FF D8 , MP3 의경우 ID 3 로 시작함.

Encase 의 View 메뉴에 가면 File Signatures 메뉴가 있다.



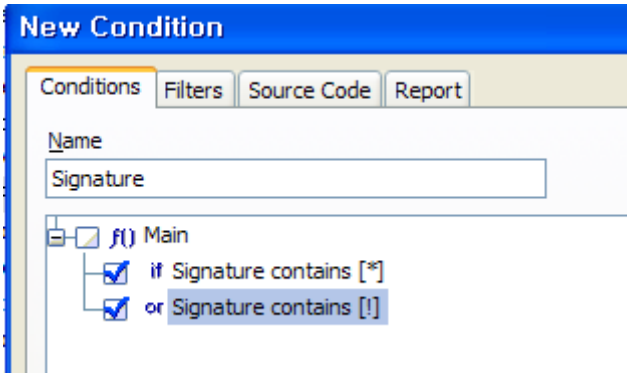
8-3. Encase에서 Search (Verify file signatures 체크) 후 결과 표시

!Bad Signature	Signiature 손상	확장자 등록, Signatre 등록 X
Match	정상적인 파일	Signature 와 확장자 일치
* [Alias]	확장자 변경	등록된 확장자, Signature 불일치 (명확한 조작)
Unknown	등록 X 파일	확장자, Signature 둘다 등록 X

8-4. Signature Search 시 필터링

Signature contains [*] : *포함한 것

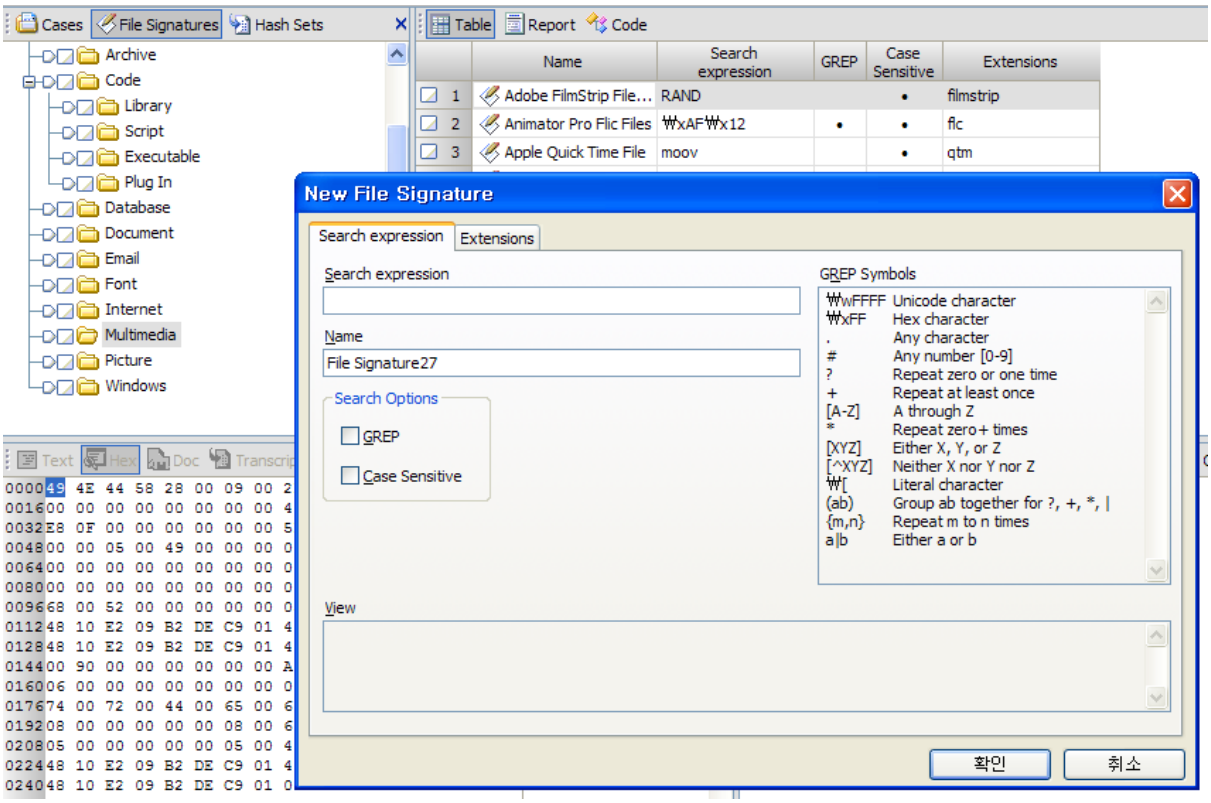
or Signature contains [!] : 복합조건 (조건을 더 만든다.)



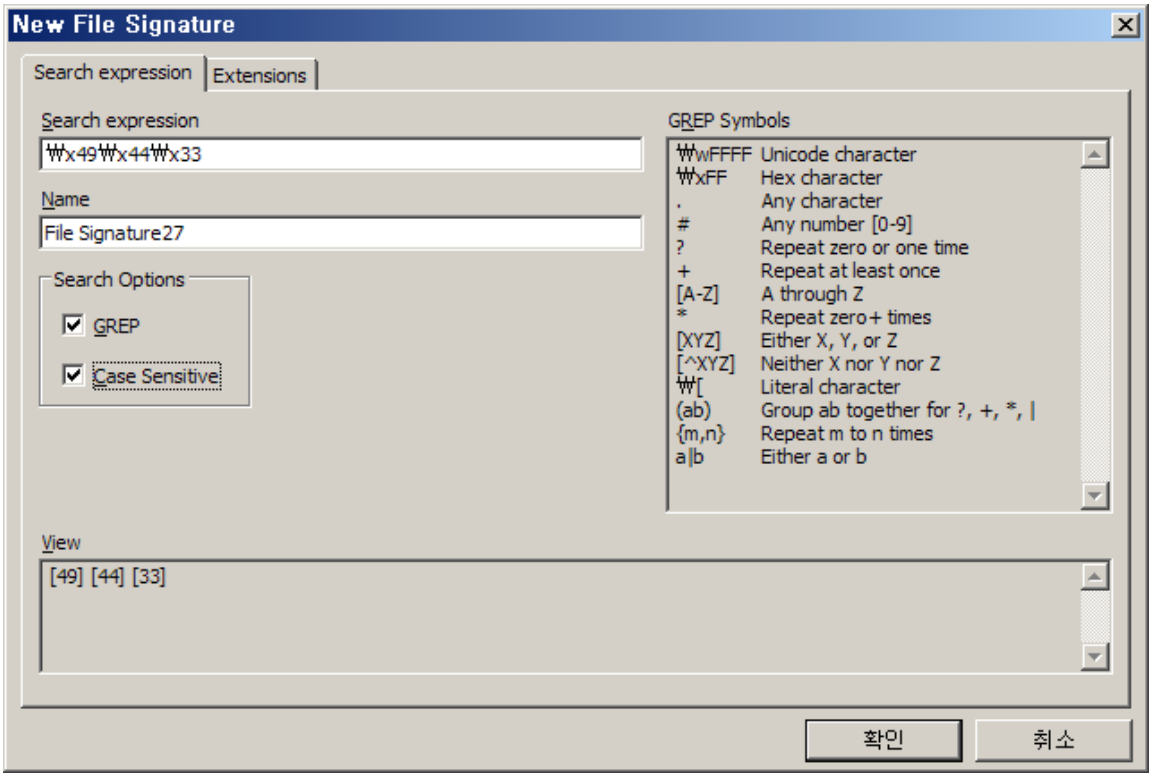
c.f) 복합조건 생성시 논리연산자가 자동으로 or 로 됨.

and 로 바꾸려면 'change logic'을 하면 된다. (우클릭 – logic)

8-5. 새로운 File Signature 등록 (Multimedia - new) : Rebuild 불필요함



예시 : Mp3 파일 등록



// Option

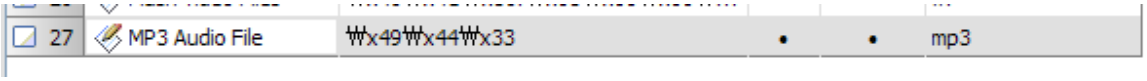
Search expression - '₩x49₩x44₩x33'

GREP - 16진수, 정규식 사용시

Case Sensitive - 대소문자 구별, Signature : ID3

Extentions : mp3

Name : mp3 Audio file



c.f) File Type 은 DB가 별도로 있어 별도 등록 필요함. Viewer는 개별설정 가능함

8-6. File Signature 분석하는 방법

파일 표준이 있을 경우 분석자료를 참고한다.

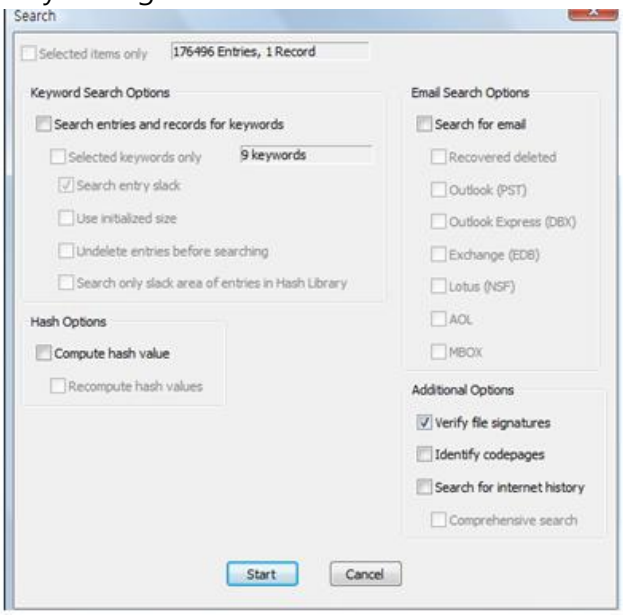
파일 샘플을 여러 개 가져다 놓고 비교 분석한다.

파일 signature를 분석해놓은 사이트를 참조한다.

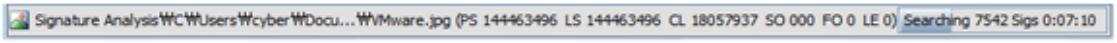
(eg. http://www.garykessler.net/library/file_sigs.html/)

8-7. File Signature 확인 실습

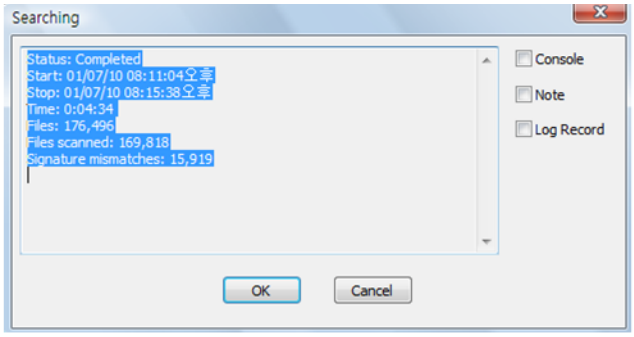
① Search – Verify file signatures



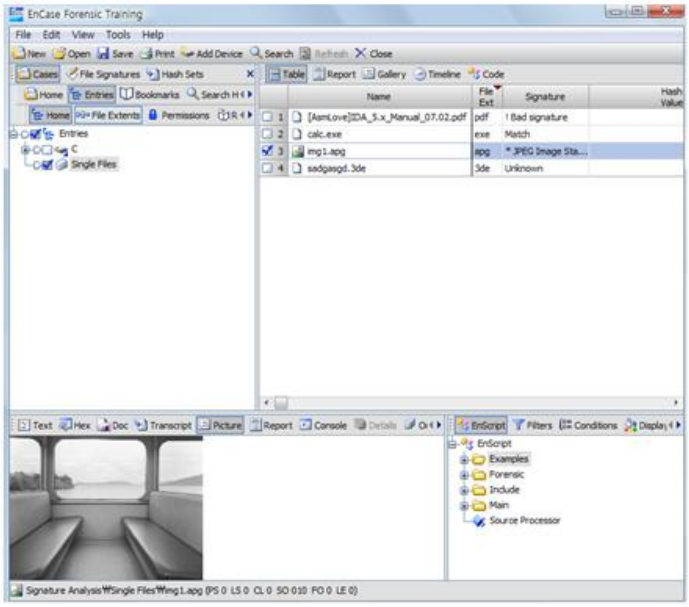
② Searching



③ Finish



④ Result



8-8. Signature 위치에 따른 파일 인식 여부

① Signature가 File의 시작이 아닌 중간에 위치한 경우 인식 여부 확인

정상적인 JPG 파일 (VMware.jpg)

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	00	01	00	60	ÿØÿà JFIF `
00000010	00	60	00	00	FF	FE	00	1F	4C	45	41	44	20	54	65	63	` ÿþ LEAD Tec
00000020	68	6E	6F	6C	6F	67	69	65	73	20	49	6E	63	2E	20	56	hnologies Inc. V
00000030	31	2E	30	31	00	FF	DB	00	84	00	05	05	05	08	05	08	1.01 ÿÛ ▮
00000040	0C	07	07	0C	0C	09	09	09	0C	0D	0C	0C	0C	0C	0D	0D	
00000050	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	
00000060	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	
00000070	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	01	05	08	08	0A	07	
00000080	0A	0C	07	07	0C	0D	0C	0A	0C	0D	0D	0D	0D	0D	0D	0D	

Signature를 중간으로 옮긴 경우 (VMware2.jpg)

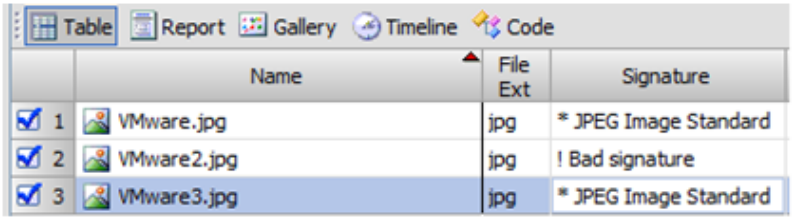
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	00	00	00	00	00	00	00	00	49	46	00	01	00	01	00	60	IF `
00000010	00	60	00	00	FF	FE	00	1F	4C	45	41	44	20	54	65	63	` ÿþ LEAD Tec
00000020	68	6E	6F	6C	6F	67	69	65	73	20	49	6E	63	2E	20	56	hnologies Inc. V
00000030	31	2E	30	31	00	FF	DB	00	84	00	05	05	05	08	05	08	1.01 ÿÛ ▮
00000040	0C	07	07	0C	0C	09	09	09	0C	0D	0C	0C	0C	0C	0D	0D	
00000050	FF	D8	FF	E0	00	10	4A	46	0D	0D	0D	0D	0D	0D	0D	0D	ÿØÿà JF
00000060	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	
00000070	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	01	05	08	08	0A	07	
00000080	0A	0C	07	07	0C	0D	0C	0A	0C	0D	0D	0D	0D	0D	0D	0D	

② File 내 여러 개의 Signature가 있는 경우 인식 여부 확인

정상적인 JPG 파일의 중간에 png 파일의 Signature 삽입 (VMware3.jpg)

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	00	01	00	60	ÿØÿà JFIF `
00000010	00	60	00	00	FF	FE	00	1F	4C	45	41	44	20	54	65	63	` ÿþ LEAD Tec
00000020	68	6E	6F	6C	6F	67	69	65	73	20	49	6E	63	2E	20	56	hnologies Inc. V
00000030	31	2E	30	31	00	FF	DB	00	84	00	05	05	05	08	05	08	1.01 ÿÛ ▮
00000040	0C	07	07	0C	0C	09	09	09	0C	0D	0C	0C	0C	0C	0D	0D	
00000050	FF	D8	FF	E0	00	10	4A	46	0D	0D	0D	0D	0D	0D	0D	0D	ÿØÿà JF
00000060	89	50	4E	47	0D	0A	1A	0A	FF	D8	FF	FE	0D	0D	0D	0D	▮PNG ÿØÿþ
00000070	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	01	05	08	08	0A	07	
00000080	0A	0C	07	07	0C	0D	0C	0A	0C	0D	0D	0D	0D	0D	0D	0D	

③ Encase에서 인식 결과



- Signature 를 중간으로 옮긴 (VMware2.jpg)의 경우 "! Bad signature"
  즉, Signature는 각 파일의 맨 앞에 위치하여야 함
- Signature 가 중복된 (VMware3.jpg)의 경우 정상적인 JPG 파일로 인식함
  즉, Signature는 각 파일의 맨 앞에 위치한 하나만 인식함

8-9. MS Office File Signature

① 2003 ver. 이하 문서 (xls, doc, ppt)

- Password 유무 상관 없이 동일한 Signature 지님 : **구분 불가능!**
  Signature : D0 CF 11 E0 A1 B1 1A E1

Word-Normal-2003.doc

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	D0	CF	11	E0	A1	B1	1A	E1	00	00	00	00	00	00	00	00	ÐÏ à¡± á
00000010	00	00	00	00	00	00	00	00	3E	00	03	00	FE	FF	09	00	>
00000020	06	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	

Excel-normal-2003.xls

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	D0	CF	11	E0	A1	B1	1A	E1	00	00	00	00	00	00	00	00	饘 胥??
00000010	00	00	00	00	00	00	00	00	3E	00	03	00	FE	FF	09	00	>
00000020	06	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	

ppt-password2-2003.ppt

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	D0	CF	11	E0	A1	B1	1A	E1	00	00	00	00	00	00	00	00	饘 胥??
00000010	00	00	00	00	00	00	00	00	3E	00	03	00	FE	FF	09	00	
00000020	06	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	

- Encase에서는 세 개 파일 모두 동일한 File Signature에 등록되어 있음
  Encase 에서는 xls, doc, ppt 모두 Compound Document File에 등록됨
  따라서 이 세 개의 확장자 중에서 서로 바꾸는 경우에도 Match 로 인식됨

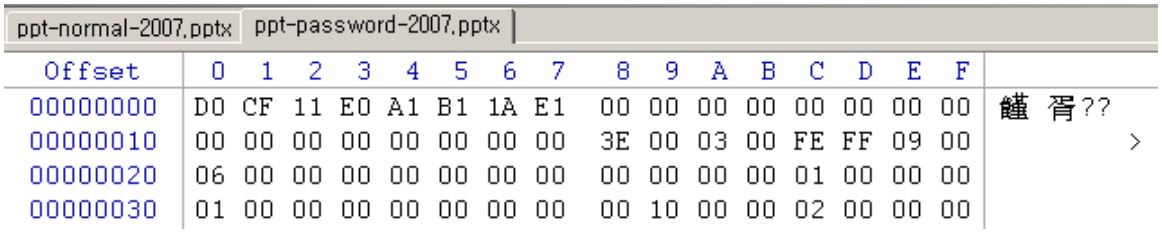
7	ppt-normal-2003.xls	xls	Match	MS Excel Spreadsheet

② 2007 ver. 문서 (xlsx, docx, pptx)

- 암호 유무에 따라 서로 다른 Signature 지님
  암호가 없는 경우 공통 Signature : 50 4B 03 04
  암호가 걸린 경우 공통 Signature : D0 CF 11 E0 A1 B1 1A E1 (2003과 동일)

ppt-normal-2007.pptx | ppt-password-2007.pptx

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	50	4B	03	04	14	00	06	00	08	00	00	00	21	00	6D	93	PK
00000010	D7	CD	17	02	00	00	DD	0D	00	00	13	00	08	02	5B	43	勒 ?
00000020	6F	6E	74	65	6E	74	5F	54	79	70	65	73	5D	2E	78	6D	ontent_Type



- Encase 의 File Signature Table 에서 확인
  암호가 없는 경우 공통 Signature : 50 4B 03 04 는 ZIP Compressed에 'zip, docx, pptx, xlsx' 확장자와 함께 등록되어 있음. 따라서 상호간 확장자를 바꾸어도 Encase는 Match로 인식함

암호가 걸린 경우 D0 CF 11 E0 A1 B1 1A E1 는 2003의 경우와 동일하게 Compound Document File에 File Signature로는 등록되어 있으나, 확장자(xlsx, docx, pptx)는 등록되어 있지 않으므로 Encase는 * (Alias) 로 인식함

③ **결론**

Office 2003 문서(암호 유무 상관 X)의 Signature
= Office 2007 문서(암호 O)의 Signature
= ZIP 파일(암호 유무 상관 X)의 Signature

9. Hash Analysis

9-1. Hash 함수

- 임의의 메시지를 입력 받아 고정된 사이즈의 결과값을 출력한다. 고정된 사이즈는 어떤 알고리즘을 사용하느냐에 따라 달라짐 MD5는 128비트이다.
- h=H(M) : Hash value 또는 Message Digest(메시지 요약)를 사용하는 이유는 대체로 메시지 사이즈보다 Hash 사이즈가 작고, 동일한 메시지에 대해 항상 같은 결과가 나오기 때문이다.
- h가 주어졌을 때 메시지 M의 내용이나 M과 관련된 어떠한 정보도 계산해 내거나 추출할 수 없는 단방향성을 지니고 있다. 즉, 역함수가 없다.
- Collision-resistance(메시지는 다른데 결과값이 같은 경우)는 거의 없다.
  eg. MD5) 1 / 340,282,366,920,938,463,463,374,607,431,768,211,456

9-2. Digital Forensic 에서 Hash 함수의 용도

- 증거의 무결성을 검증하기 위해 사용한다.
  디지털 증거는 조금이라도 조작되었을 경우 크게 다른 결과를 가져올 수 있으므로 무결성을 검증하는 것이 매우 중요하다.

9-3. Hash 분석의 방식

① Negative Hash analysis

i. 의의
   - Known-good hash set을 이용하는 hash 분석 방법

ii. 조사방법
   - 증거이미지에 포함된 파일의 hash값을 모두 발생
   - Known-good hash set과 비교하여 매칭되는 파일을 조사 목록에서 제거

② Positive Hash analysis

i. 의의
   - Known-bad hash set을 이용하는 hash 분석 방법

ii. 조사방법
   - 증거이미지에 포함된 파일의 hash값을 모두 발생
   - Known-bad hash set과 비교하여 매칭되는 파일을 모두 조사

9-4. Hash set

① 의의 : 선택한 파일의 해쉬값을 모아놓은 DB

② 종류

i. Known-good Hash set

- 정상적인 파일의 hash 값 집합
- 조사할 필요가 없는 파일의 hash값의 Data Base
- OS의 버전이 다양하기 때문에 만들기 힘들다.
- c.f) NSRL(National Software Reference Library) : 미국의 NIST 산하에서 프로젝트로 만든 known-good hash set으로서, 우리나라 환경에는 잘 맞지 않지만 매우 유용하게 사용될 수 있다. 우리나라도 환경에 맞는 known-good hash set이 필요하다. (참조 : http://www.nsrl.nist.gov/)

ii. Known-bad Hash set

- 해킹에 사용된 툴, 악성코드 등 조사할 필요가 있는 파일의 hash값의 Data Base

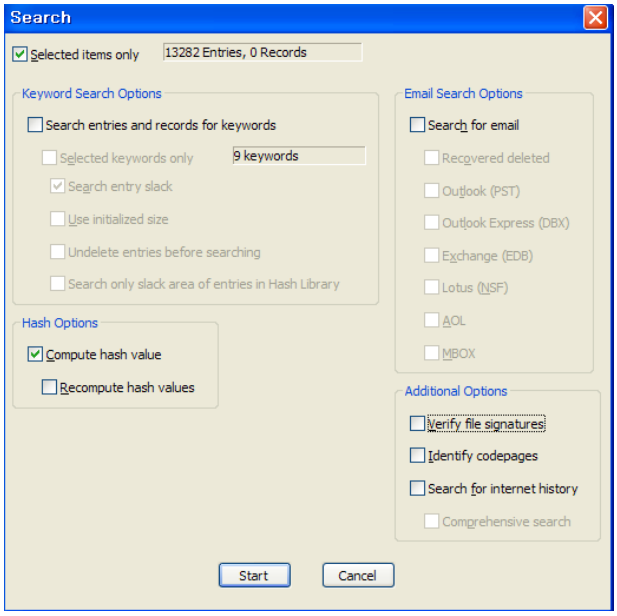
9-5. Empty파일(size가 0인 파일) 제거

- 컴퓨터 상에는 size가 0인 파일이 많은데, 이는 특정 파일이 특정 프로그램의 존재여부와 관련된 경우와 같이 조사가 필요한 특별한 경우가 아니면 조사할 필요가 없다.
- Encase는 기본적으로 비어있는 파일에 대한 Hash값을 가지고 있다.
- Hash 분석을 하면 기본적으로 Hash set에서 Empty file으로 분류한다. Category는 known으로 분류되어 있다.
- c.f) Size가 0인 파일 만들기 : (Windows) Fsutil file createnew [파일명] [size]
  (Linux) touch [파일명]

9-6. **Hash set 생성 실습** (Hash set은 생성 후 Rebuild 필요)

① View - hash set 에서 hash sets 탭을 만든다.

② Cases entry에서 해시 값 만들 파일들을 선택한 후 Search 기능 이용하여 hash값 만들어낸다.



// Selected items only : 선택된 파일만

Compute hash value : hash값 생성

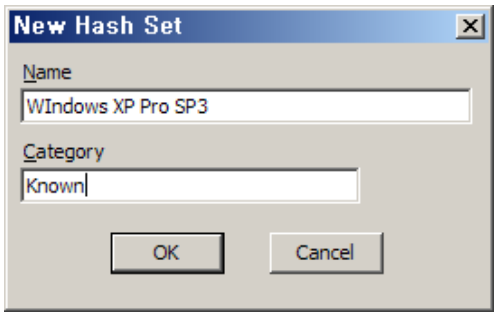
	References	Physical Location	Physical Sector	Evidence File	File Identifier	Code Page	Hash Value
☐ 60	0	42,414,080	82,840	C	160	0	
☑ 61	0	42,774,528	83,544	C	173	0	c3200506fb212a0f4fb736a80e646c40
☑ 62	0	42,778,624	83,552	C	174	0	c50779f711d96319de3d045ffbbd78cf
☑ 63	0	42,803,200	83,600	C	175	0	381f44464add019d24b17731544ea8bd
☐ 64	0	42,901,504	83,792	C	176	0	
☐ 65	0	43,024,384	84,032	C	180	0	
☐ 66	0	43,118,592	84,216	C	187	0	
☑ 67	0	43,130,880	84,240	C	188	0	56c5b179fe3308b655eb6208c3256fec
☑ 68	0	43,528,192	85,016	C	199	0	9d3df937038bc3e006643276be23f8b0
☐ 69	0	44,851,200	87,600	C	208	0	
☐ 70	0	44,879,872	87,656	C	210	0	

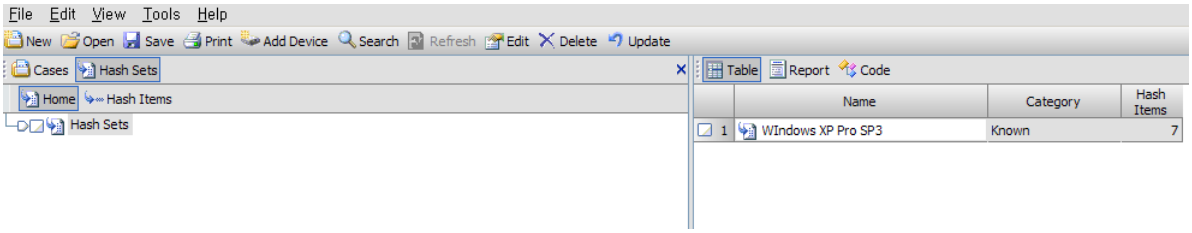
// 다음과 같이 선택된 파일에 한해서 Hash Value가 도출된 것을 볼 수 있다.

③ Hash set 탭에서 New 기능을 통해 Hash set 추가
(도출된 Hash값을 Known으로 등록)

i. New hash set에서 Category는 Notable, Known 두 가지만 사용하는 것이 좋다. 이는 Guidance 社의 권고사항으로, Enscript가 이 두 가지 Hash set을 가정하고 작성된 것이 많기 때문에 이를 활용하기 위함이다.

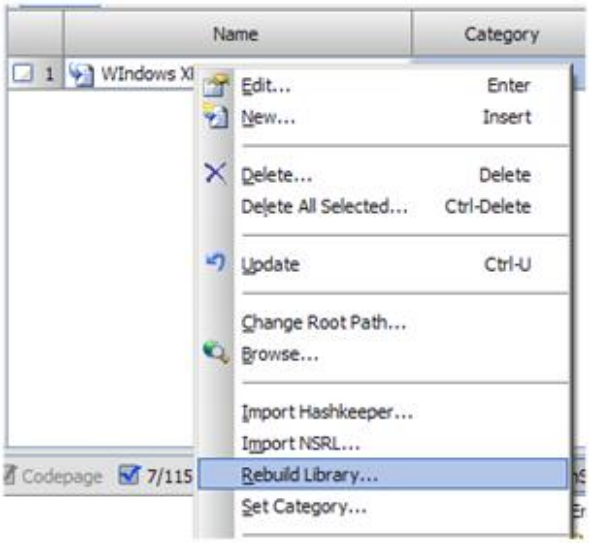
ii. Known : Known-good Hash set
Notable : Known-bad Hash set





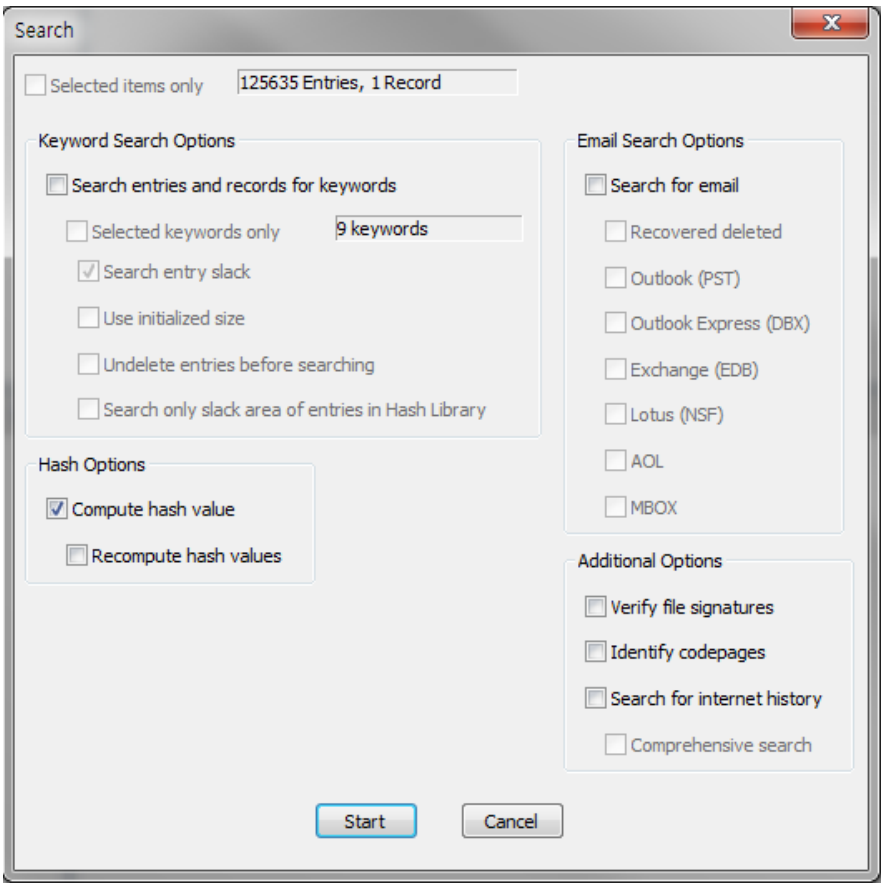
// Hash Set이 만들어졌다.
Encase가 설치된 폴더의 Hash Sets 폴더에 저장되며, 복사 가능하다.

④ Rebuild 필요
Hash set 생성 후 당장은 사용이 안 된다. 해당 Hash set을 체크한 후 Rebuild 하거나(우클릭- Rebuild Library) 다시 껐다 켜야 한다.



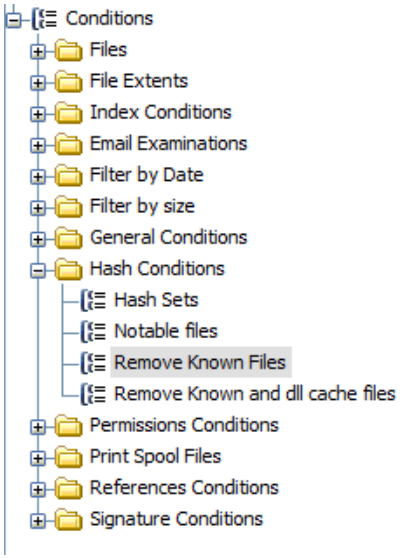
9-7. Hash set을 이용한 조사 실습 (Conditions 활용)

① 필요한 Hash set 선택 후, Search- Compute hash value 체크 -> Search!

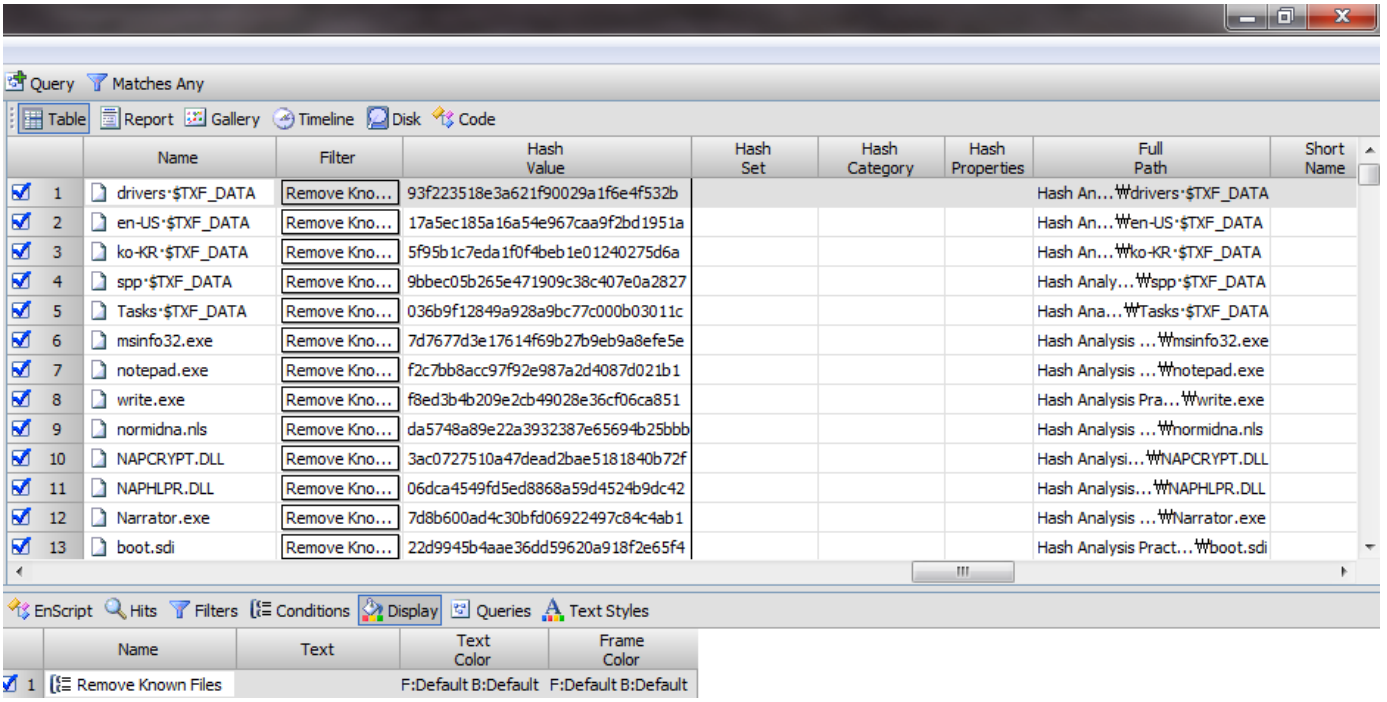


② 우측 하단의 필터 pane을 이용하여 조사 파일의 양을 줄일 수 있다. 예를 들어, Hash category가 known인 것은 제거할 수 있다.

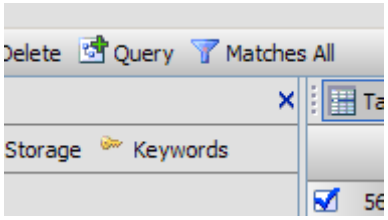
Condition : 해당 조건에 맞는 데이터만 보는 것이 가능하다.
(Enscript로 이루어짐. C++ 문법과 유사함)



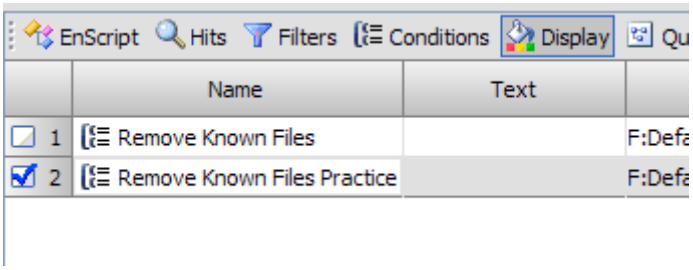
③ Known Files 는 제외하고 필터링한 결과



④ Query 버튼을 누르면 조건이 적용된다.

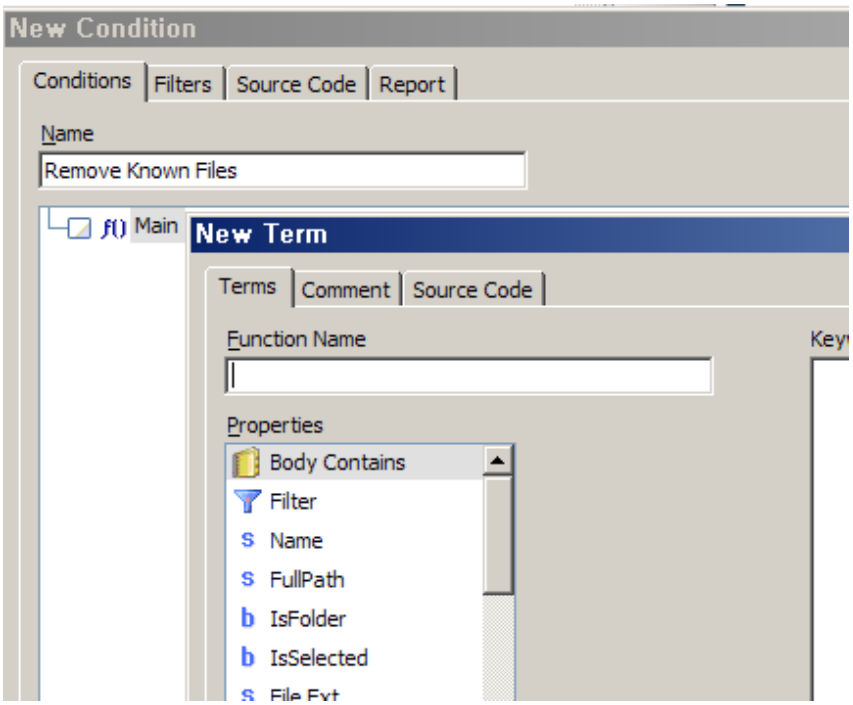


선택적 적용은 Display 탭에서 체크한다.



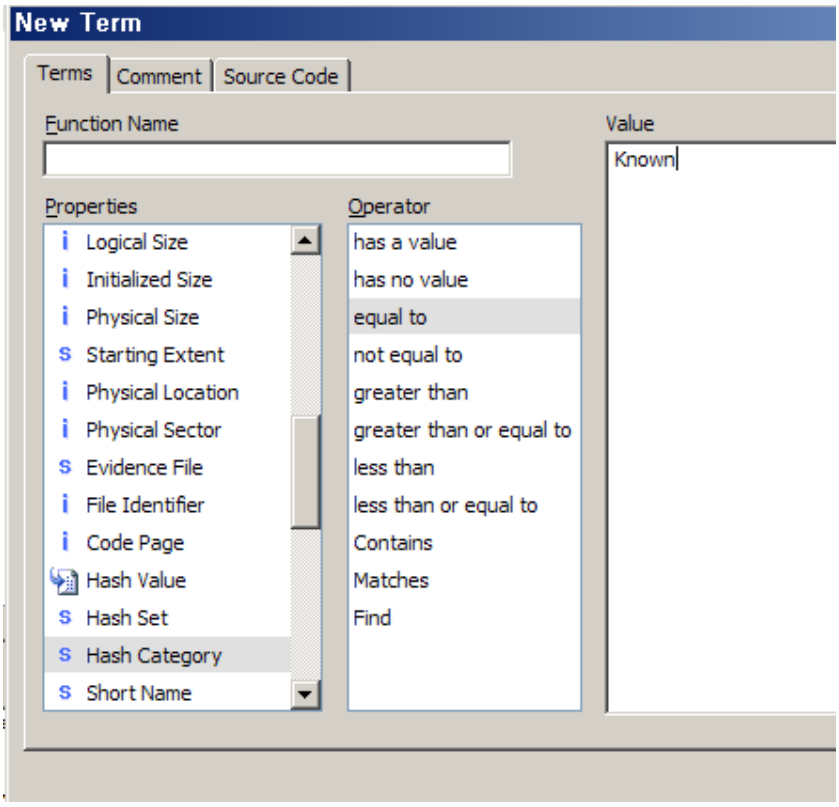
⑤ 참고 : Hash 카테고리 Known 인 파일을 제외시키는 Condition 만들기

i. New Condition (Name : Remove Known Files)

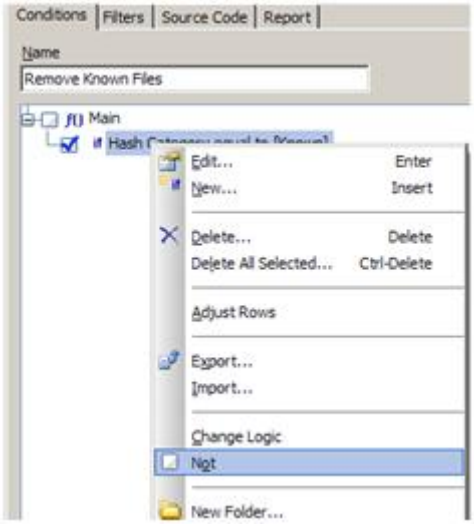


ii. 함수 추가 New

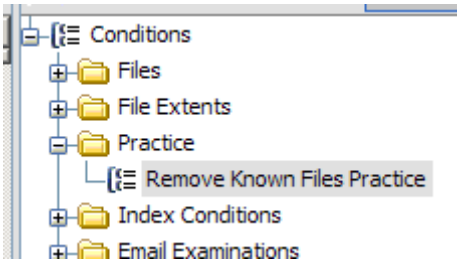
(Properties : Hash Category, Operator : equal to, Value : Known)



iii. Not 적용



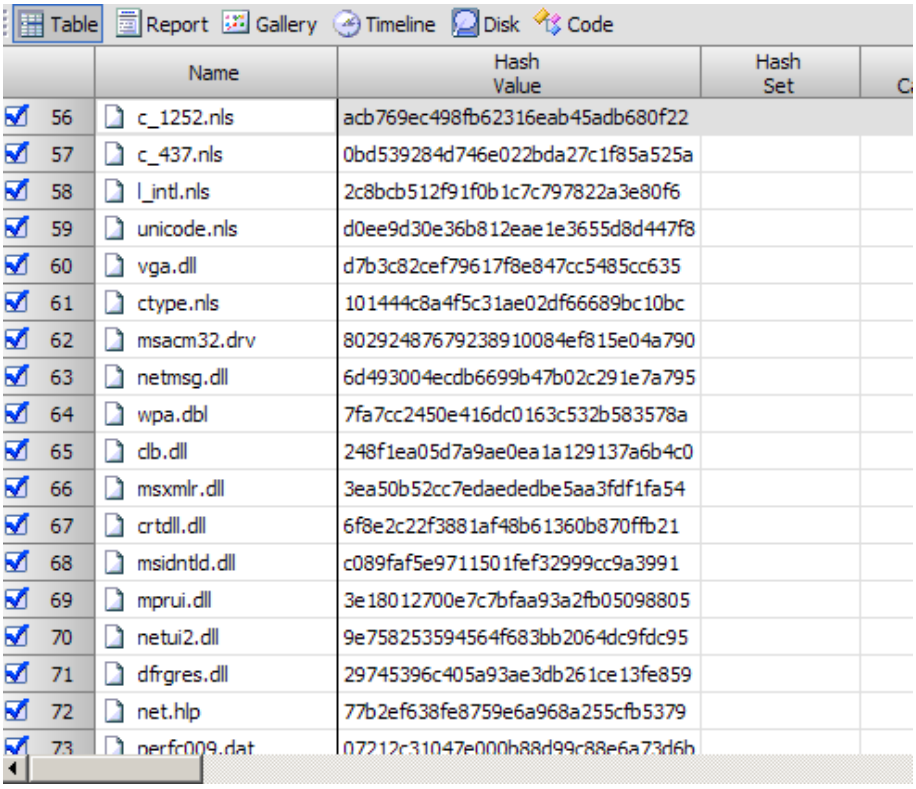
iv. 직접 입력하여 생성된 Condition



v. 이를 Run 해보면, Known Files는 제외되는 것을 볼 수 있다.

56	bootvid.dll	cc306bf581446d5e443eae5b3bb900f0	WIndows XP P...	Known
57	kdcom.dll	945fbb881ae927a44dfd96440f2f4f44	WIndows XP P...	Known
58	c_1252.nls	acb769ec498fb62316eab45adb680f22		
59	c_437.nls	0bd539284d746e022bda27c1f85a525a		
60	l_intl.nls	2c8bcb512f91f0b1c7c797822a3e80f6		
61	lz32.dll	c3200506fb212a0f4fb736a80e646c40	WIndows XP P...	Known
62	olesvr32.dll	c50779f711d96319de3d045ffbbd78cf	WIndows XP P...	Known
63	olethk32.dll	381f44464add019d24b17731544ea8bd	WIndows XP P...	Known
64	unicode.nls	d0ee9d30e36b812eae1e3655d8d447f8		
65	vga.dll	d7b3c82cef79617f8e847cc5485cc635		
66	ctype.nls	101444c8a4f5c31ae02df66689bc10bc		
67	kbdus.dll	56c5b179fe3308b655eb6208c3256fec	WIndows XP P...	Known
68	netevent.dll	9d3df937038bc3e006643276be23f8b0	WIndows XP P...	Known
69	msacm32.drv	80292487679238910084ef815e04a790		
70	netmsg.dll	6d493004ecdb6699b47b02c291e7a795		

// 적용 전



	Name	Hash Value	Hash Set	C
56	c_1252.nls	acb769ec498fb62316eab45adb680f22		
57	c_437.nls	0bd539284d746e022bda27c1f85a525a		
58	l_intl.nls	2c8bcb512f91f0b1c7c797822a3e80f6		
59	unicode.nls	d0ee9d30e36b812eae1e3655d8d447f8		
60	vga.dll	d7b3c82cef79617f8e847cc5485cc635		
61	ctype.nls	101444c8a4f5c31ae02df66689bc10bc		
62	msacm32.drv	80292487679238910084ef815e04a790		
63	netmsg.dll	6d493004ecdb6699b47b02c291e7a795		
64	wpa.dbl	7fa7cc2450e416dc0163c532b583578a		
65	clb.dll	248f1ea05d7a9ae0ea1a129137a6b4c0		
66	msxmlr.dll	3ea50b52cc7edaededbe5aa3fdf1fa54		
67	crtdll.dll	6f8e2c22f3881af48b61360b870ffb21		
68	msidntld.dll	c089faf5e9711501fef32999cc9a3991		
69	mprui.dll	3e18012700e7c7bfaa93a2fb05098805		
70	netui2.dll	9e758253594564f683bb2064dc9fdc95		
71	dfrgres.dll	29745396c405a93ae3db261ce13fe859		
72	net.hlp	77b2ef638fe8759e6a968a255cfb5379		
73	perfc009.dat	07212c31047e000b88d99c88e6a73d6b		

// 적용 후 : Known Files 가 보이지 않음

10. Shortcut Analysis

10-1. Link Files 분석 개요

- 바탕화면, 최근문서, 시작메뉴, Send to 폴더 등에는 shortcut 형태의 증거들이 있다. Shortcut 파일들은 타겟 파일(어플리케이션, 디렉토리, 데이터 파일, 프린터 등)에 대한 정보를 가지고 있다. 이를 분석하여 필요한 정보를 추출해낼 수 있다.

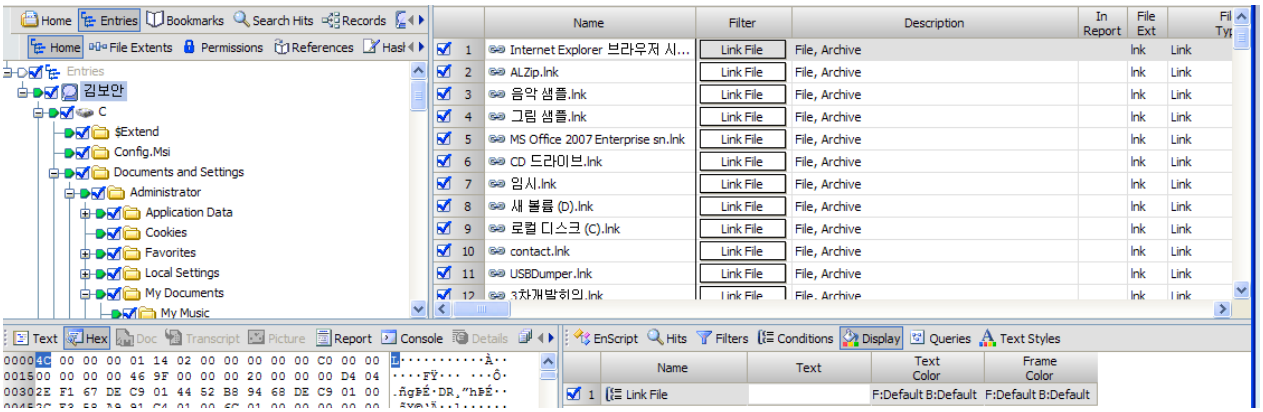
10-2. Link File 생성 시기

- 사용자가 의도적으로 생성시
- 프로그램 설치시
- 쉘(explorer)이나 OS가 파일 실행시

10-3. Link File 골라내기 (Condition 기능 이용)

Conditions - Extention - Match – lnk

Conditions - FileType - equal to - LINK 로 등록하여 확인

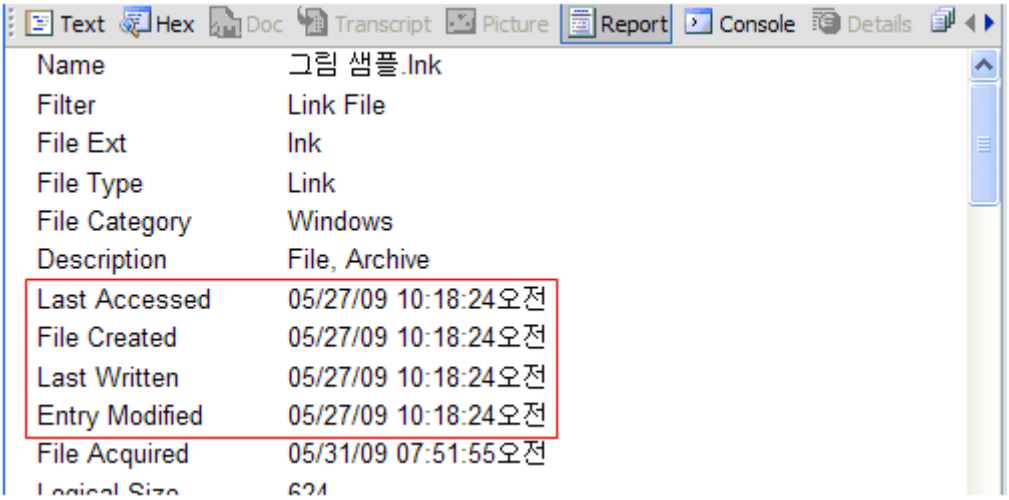


10-4. **Link File 활용** (Link File의 MAC Time 정보)

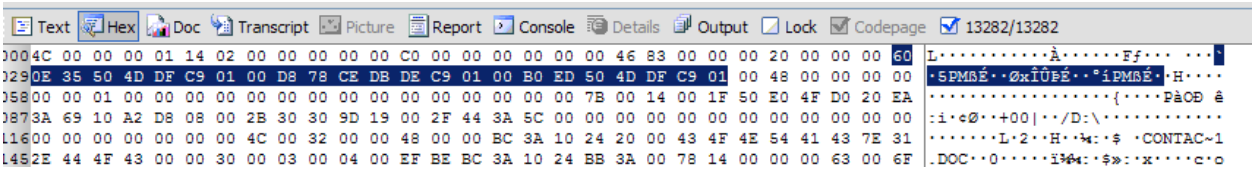
① Link 파일 안에는 Link file 생성 당시의 타겟 파일의 MAC Time 정보(CAM 정보)가 기록되어 있다.

a. Windows : Created, accessed, modify

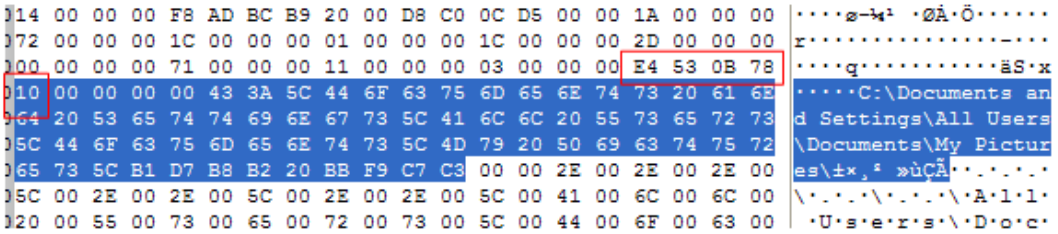
b. Linux : Modify, accessed, changed



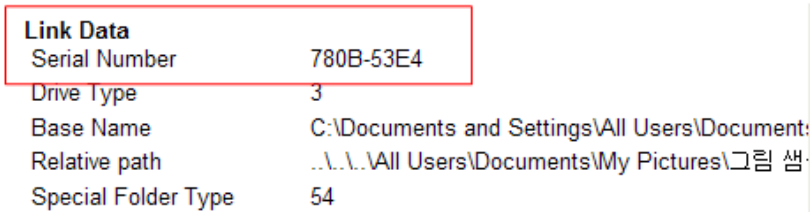
② Windows 에서 개개의 날짜/시간 정보는 8bytes 로, Link File 에서 MAC Time 정보는 Offset 28 번에서부터 24bytes 에 표현된다.
File Created Time (타겟 파일 생성 시각)과 Last Written (마지막 수정 시각)이 몇 초 차이나는 경우에는 해당 파일의 '다운로드' 가능성을 의심해볼 수 있으며, 이 두 시각이 동일할 경우 '복사' 가능성이 있다.



③ USB 를 PC 에 꽂아 어떤 파일을 설치 또는 사용하였는지 여부를 확인하여 인지사실에 대한 증거로 사용하려는 경우, Link File 의 시간정보를 가지는 Volume Serial Number (타겟 파일의 위치를 나타내는 값 앞 10 을 찾고 그 전 4 바이트)와 USB 파일시스템 내부의 Volume Serial Number 를 비교하여 찾을 수 있다. 해당 Target 파일이 들어있는 파일시스템의 Volume Serial Number 가 Link File 에 부여되기 때문이다.

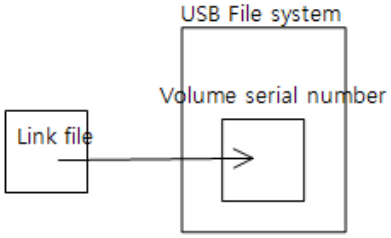


// Hex '10'을 찾은 후, 그 앞 4 bytes 가 Volume Serial Number (Unicode)



// Encase 의 Report 탭으로 확인한 Link File 의 Volume Serial Number

이를 USB 파일시스템 내부의 Volume Serial Number 와 동일한지 비교하여 추출한다.



11. 참고문헌

- File System Forensic Analysis, Carrier 저, Addison-Wesley, 2005.01.01
- Encase Computer Forensics--the Official Ence (Paperback / 2nd Ed.) (Encase Certified Examiner Study Guide), Bunting, Steve 저, John Wiley & Sons Inc, 2007.09.12
- Encase Forensic Academic Program Students Guide, Guidance

12. 도움주신 분들

- 이지스원 시큐리티 보안팀장 김 태 일
- 이지스원 시큐리티 연구원 이명수, 주한익
- 경찰수사연수원 사이버 교수 유 현
- 국립 경찰 대학 사이버 교수 장 윤 식
- 육군 사관 학교 사이버 교관 유 정 호

---

[i] **http://forensic.kr/tc/ | CharSyam's Blog**

[ii] **설명참조 : http://forensic.kr/tc/ | CharSyam's Blog**
그림참조 : http://ccibomb.tistory.com | Ccibomb's Blog

[iii] **Derrick J. Farmer. "A FORENSIC ANALYSIS OF THE WINDOWS REGISTRY", 2008**
: http://www.eptuners.com/forensics/Index.htm