

NTFS - Part 1 (보고서)

날짜	@2024년 2월 11일
사람	ohnahee

Chapter 1. 디지털 조사 기초

1.1. 디지털 조사와 증거

1.2. 디지털 범죄 현장 조사 절차

[시스템 보존 단계](#)

[증거 탐색 단계](#)

[사건의 재구성 단계](#)

[일반지침](#)

1.3. 데이터 분석

[분석 유형](#)

[필수데이터/부가데이터](#)

1.4. 툴킷 개요

Chapter 2. 컴퓨터 기초

2.1. 데이터 구성

[2진수/10진수/16진수](#)

[데이터 크기](#)

[스트링과 문자 인코딩](#)

[데이터 구조체](#)

[플래그 값](#)

2.2. 부팅과정

[중앙처리장치와 기계어](#)

[부트코드위치](#)

2.3. 하드디스크 기술

[하드디스크 위치정보와 내부구조](#)

[ATA/IDE *인터페이스](#)

[섹터주소유형](#)

[인터페이스 표준](#)

[디스크 명령어](#)

[하드디스크 패스워드](#)

[HPA\(Host Protected Area\)](#)

[DCO \(Device Configuration Overlay\)](#)

[직렬 ATA](#)

[BIOS와 직접접근](#)

[SCSI 드라이브](#)

[커넥터 타입](#)

Chapter 3. 하드디스크 데이터 수집

3.1. 소개

[수집 절차](#)

[데이터 수집 계층](#)

3.2. 원본 데이터 읽기

[직접접근과 BIOS 접근](#)

[정적수집과 동적수집](#)

[오류 처리](#)

[숨겨진 영역 또한 수집](#)

[하드웨어 쓰기방지 장치](#)

[소프트웨어 쓰기 방지 장치](#)

3.3. 출력 데이터 쓰기

[목적지](#)

[이미지 파일 형식](#)
[이미지 파일 압축](#)
[네트워크 기반 수집](#)
[무결성해시](#)
[3.4. dd를 이용한 실습](#)
[입력소스](#)
[HPA](#)
[출력 목적지](#)
[오류처리](#)
[암호해시](#)

Chapter 1. 디지털 조사 기초

1.1. 디지털 조사와 증거



책에서 사용하는 정의와 근거

- 디지털 조사
 - 사건과 관련된 여러 의문에 대해 가설을 세우고 가설을 검증해 나가는 과정
- 디지털 증거
 - 가설을 반박하거나 뒷받침하는 신뢰할 만한 정보를 포함한 디지털 객체
- 증거
 - 법률적 관점
 - 분석적 관점 (이 책의 관점)
- 포렌식
 - 수사에서 과학이나 기술사용과 관련이 있고 법정에서 채택되는 증거나 사실을 증명하는 것
- 디지털 포렌식 조사
 - 디지털 객체를 조사하기 위해 과학과 기술을 사용하는 절차
 - 법정에서 사용할 이론을 세우고 검증하는 과정

- 디지털 조사
 - 법정 요구 사항이 아닌 기술에만 집중

1.2. 디지털 범죄 현장 조사 절차



디지털 범죄 현장 조사의 주요 3단계

시스템 보존 단계

- 디지털 범죄 현장 상태를 그대로 유지하는 것
- 덮어써져서 지워지는 증거를 줄이기 위함
 - 데이터 보존 기술
 - 쓰기 방지 장치 사용
 - 암호해시계산

증거 탐색 단계

- 발견 됐던 지점을 조사하는 것
- 탐색기술 (파일시스템과 파일 내부에서 이루어짐)
 - 이름, 이름패턴, 키워드를 통해
 - 마지막 접근시간, 수정시간 같은 임시데이터

사건의 재구성 단계

- 위에서 얻어낸 증거를 통해 사건이 발생함을 판단

일반지침

- PICL 에 유의

- 보존 (Preservation)
- 격리 (Isolation)
 - 악성프로그램으로부터 격리
- 상호연관 (Correlation)
 - 독립적인 요인들과 데이터를 상호연관시켜 분석
- 로깅 (Logging)
 - 조사와 관련된 행동들을 기록하고 문서화 → 변화나 결론을 내기위해

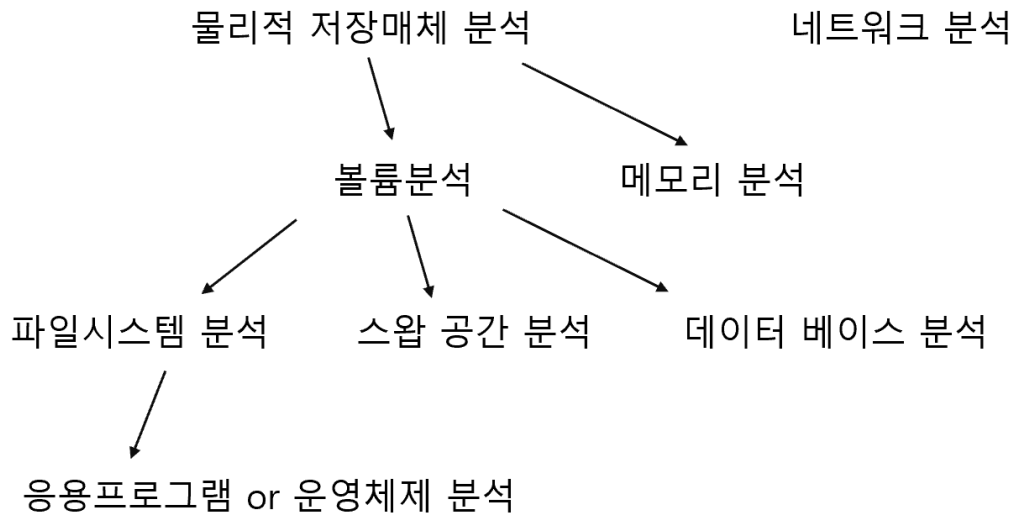
1.3. 데이터 분석



1. 디지털 증거를 찾을 수 있는 곳 범위 좁히기
2. 어떤 데이터를 신뢰해야하는지

분석 유형

- 디지털 데이터는 설계된 객체
- 디지털 장치 저장시스템은 가변적이고 계층적이다



디지털 데이터 설계의 분석계층

- 물리적 저장 매체 (하드디스크)
 - 볼륨 (분할/결합)
 - 파일시스템 : 파일 내용, 데이터 조각, 파일과 연관된 메타데이터들
 - 응용프로그램 계층
 - 운영체제
 - 데이터 베이스 포함
 - 스왑공간 (숨겨진 영역)
- 네트워크 통신장치 분석

필수데이터/부가데이터

*2, 3부에서 자세히 구분

- 각 계층에 모든 데이터에는 구조체가 존재한다
- 계층에서 핵심적인 목적을 제공하기 위해 모든 구조체가 필요하지 않음
- 파일 시스템 계층은 빈 볼륨을 구성한 후 데이터를 저장하고 읽도록 한다
 - 파일내용과 파일명을 서로 연관시키는 것이 필요
 - 파일명, 파일내용, 디스크 위치 필요 → 필수데이터
- 주소를 신뢰한다고 해서 실제 내용이 신뢰되지는 않음

1.4. 툴킷 개요



모든 도구는 보존 또는 탐색에 초점이 맞추어져있음

- Encase : 윈도우 기반 가장 보편적
- Forensic Toolkit (FTK) : 윈도우 기반
- ProDiscover : 윈도우 기반
- SMART : 리눅스 기반
- The Sleuth Kit / Autopsy : 유닉스기반의 툴 TSK, 그리고 그 결과를 그래픽으로 보여주는 Autopsy

Chapter 2. 컴퓨터 기초

2.1. 데이터 구성



데이터 기본원리, 2진수와 16진수, 데이터 크기, 엔디안 순서와 데이터 구조
→ 데이터가 어떤 형태로 저장되어 있는지 이해

2진수/10진수/16진수

- 2진수
 - 컴퓨터가 동작하는 방식
 - 0과 1뿐
 - 0또는 1은 비트 / 0과 1이 8개 모이면 바이트

- 10진수
 - 10개인 기호들의 연속
 - 각 기호는 한 개의 값을 가짐
- 16진수
 - 16개인 기호들의 연속
 - 16진수 앞에 0x가 붙음

데이터 크기

- 디지털 데이터를 저장하기 위해서는 저장 장치 위치를 데이터에 할당 해주어야 함
- 데이터가 저장되는 가장 작은 크기는 1바이트 = 256개의 수 표현가능
- 더 큰 수를 저장하기 위해 몇 개의 바이트를 그룹 지음
 - 보통 바이트 그룹의 크기는 2, 4, 8바이트

컴퓨터가 여러 바이트의 값을 구성하는 방법

빅 엔디안	첫 번째 저장 바이트를 최상위 바이트에 둠
리틀 엔디안	첫 번째 저장 바이트를 최하위 바이트에 둠

스트링과 문자 인코딩

- 문자는 1바이트 당 한 문자씩 저장되기 때문에 엔디안이 적용되지 않는다
- 단어나 문장 내 첫 번째 문자는 항상 할당된 첫 번째 바이트에 위치한다
- 한 단어나 문장에 연속된 바이트 = 스트링 (String)
 - 0x00인 NULL 기호로 끝난다
- 컴퓨터는 숫자로 이루어져 있기 때문에 각각의 글자를 숫자로 표현해서 집어넣는다
 - 아스키코드 : 알파벳을 숫자로 지정해놓은 규칙

제어 문자		공백 문자		구두점		숫자		알파벳			
10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진	문자
0	0x00	NUL	32	0x20	SP	64	0x40	@	96	0x60	`
1	0x01	SOH	33	0x21	!	65	0x41	A	97	0x61	a
2	0x02	STX	34	0x22	"	66	0x42	B	98	0x62	b
3	0x03	ETX	35	0x23	#	67	0x43	C	99	0x63	c
4	0x04	EOT	36	0x24	\$	68	0x44	D	100	0x64	d
5	0x05	ENQ	37	0x25	%	69	0x45	E	101	0x65	e
6	0x06	ACK	38	0x26	&	70	0x46	F	102	0x66	f
7	0x07	BEL	39	0x27	'	71	0x47	G	103	0x67	g
8	0x08	BS	40	0x28	(72	0x48	H	104	0x68	h
9	0x09	HT	41	0x29)	73	0x49	I	105	0x69	i
10	0x0A	LF	42	0x2A	*	74	0x4A	J	106	0x6A	j
11	0x0B	VT	43	0x2B	+	75	0x4B	K	107	0x6B	k
12	0x0C	FF	44	0x2C	,	76	0x4C	L	108	0x6C	l
13	0x0D	CR	45	0x2D	-	77	0x4D	M	109	0x6D	m
14	0x0E	SO	46	0x2E	.	78	0x4E	N	110	0x6E	n
15	0x0F	SI	47	0x2F	/	79	0x4F	O	111	0x6F	o
16	0x10	DLE	48	0x30	0	80	0x50	P	112	0x70	p
17	0x11	DC1	49	0x31	1	81	0x51	Q	113	0x71	q
18	0x12	DC2	50	0x32	2	82	0x52	R	114	0x72	r
19	0x13	DC3	51	0x33	3	83	0x53	S	115	0x73	s
20	0x14	DC4	52	0x34	4	84	0x54	T	116	0x74	t
21	0x15	NAK	53	0x35	5	85	0x55	U	117	0x75	u
22	0x16	SYN	54	0x36	6	86	0x56	V	118	0x76	v
23	0x17	ETB	55	0x37	7	87	0x57	W	119	0x77	w
24	0x18	CAN	56	0x38	8	88	0x58	X	120	0x78	x
25	0x19	EM	57	0x39	9	89	0x59	Y	121	0x79	y
26	0x1A	SUB	58	0x3A	:	90	0x5A	Z	122	0x7A	z
27	0x1B	ESC	59	0x3B	;	91	0x5B	[123	0x7B	{
28	0x1C	FS	60	0x3C	<	92	0x5C	₩	124	0x7C	
29	0x1D	GS	61	0x3D	=	93	0x5D]	125	0x7D	}
30	0x1E	RS	62	0x3E	>	94	0x5E	^	126	0x7E	~
31	0x1F	US	63	0x3F	?	95	0x5F	_	127	0x7F	DEL

- 아스키방식 말고도 다른 방식들이 있다
 - 유니코드 : 저장해야될 기호가 1바이트보다 더 클때 사용
 - 유니코드 문자를 저장하는 세 가지 방식
 - UTF-32 : 각 문자를 4바이트 단위로 표시
 - UTF-16 : 가장 많이 사용된 문자를 2바이트, 나머지를 4바이트
 - UTF-8 : 한 문자의 저장을 위해 1, 2, 4 바이트 사용

데이터 구조체

- 데이터 구조체는 데이터가 어떻게 설계되어있는지 설명하는 템플릿, 지도 같은역할
- 데이터를 보내야 하는 편지라고 생각한다면 목적지를 알아야함

- 0-1 (2바이트) : 번지
- 2-32 (30바이트) : 거리 (아스키값)

EX) 1번지 Main St 에 가야하는 편지 (데이터) 가 있다고 하자

저장 공간 바이트에서

- 1) 0-1사이에 번지수 1을
- 2) 2-9바이트에는 Main St 를 각 문자의 아스키 값이 무엇인지를 확인해서 해당 위치에 쓴다
- 3) 나머지 바이트들은 0으로 설정 (필수X)

- 데이터 구조체의 바이트 위치는 할당한 저장 위치에서 상대적이다
- 번지 수의 바이트들의 순서는 컴퓨터의 엔디안 순서에 따라 다르다

- 저장장치에서 데이터를 읽을 경우는 데이터 구조체 시작위치를 확인한 후 참조한다

EX) 이전 예에서 썼던 데이터 확인 시

👉 00000000 : 0100 4d61 696e 2053 742e 0000 0000 0000 ..MainSt.....
00000016 : 0000 0000 0000 0000 0000 0000 0000 0000
00000032 : 1900 536f 7574 5374 2e00 0000 0000 0000 ..South St....
00000048 : 0000 0000 0000 0000 0000 0000 0000 0000

- 1) 저장 장치에서 데이터 구조체가 어디에서 시작하는지 확인
- 2) 이전에 확인한 데이터 구조 템플릿을 적용
- 3) 왼쪽 열은 10진수로 나타낸 바이트 오프셋
8개의 중간열들은 16진수 값을 갖는 데이터의 16개 바이트
마지막 열은 데이터인 ASCII (' ' 은 아스키로 표현할 수 없는 값)
- 4) 각 번지수는 32바이트
0-1 (2바이트) 번호
2-31 (30바이트) 거리

플래그 값

- 일부 데이터를 1이나 0으로 표현해 그것들의 존재여부를 확인

→ 1비트만 필요한 곳에 8비트(바이트)를 할당하기 때문에 공간 낭비이긴 함

- 플래그 보는 법
 - 플래그 공간에 0x61 저장되어있을 때
 - 0x61 = 0110 0001
 - 이때 최하위 비트가 1이면 참으로 본다
 - 2진수로 변환 후 제일 마지막 비트를 보면 됨

2.2. 부팅과정



1. 컴퓨터 동작을 위한 필수데이터
2. 필수데이터를 가져오기 위한 명령어인 부트코드
3. 부팅과정
4. 부트코드 위치

중앙처리장치와 기계어

- CPU (Center Processing Units, 중앙처리장치)
 - 컴퓨터의 핵심
 - 명령어의 입력 필요
 - 메모리에서 명령어를 가져오는 역할 수행
 - 디스크에서 기계어 (명령어)
 - 각 기계어 명령어의 길이는 일정 수 바이트
 - 첫번째 몇 바이트 → opcode : 명령어 유형
- EX) opcode 0xB400 = Mov AH 00
- 즉 AH 레지스터로 Mov (이동) 하라는 의미

부트코드위치

- 부트코드 : **CPU의 명령어들을 입력하는 장치**
- 대부분 시스템들은 모든 하드웨어를 깨워서 실행한 후 운영체제 또는 다른 소프트웨어 실행
- 부트코드는 모든 볼륨과 파일 시스템의 특정 영역에 저장되어 있음
 - 모든 볼륨에서 필요로 하지는 않음

- 1) 전원을 키면 CPU 는 ROM과 같은 메모리 내 특정 위치로 부터 명령어를 읽어옴
- 2) 그 명령어들은 시스템 하드웨어를 조사하고 설정함
- 3) CPU는 추가의 부트코드가 있는 장치를 찾음
- 4) 장치를 찾으면 부트코드 실행
- 5) 부트코드는 특정 운영체제의 위치를 찾고 적재함

- 윈도우 부팅 절차
 - 1) 전원 ON
 - 2) CPU는 BIOS 로 부터 명령어를 읽어들임
 - 3) 하드디스크, 드라이브, 다른 하드웨어의 위치를 알아냄
 - 4) BIOS는 설정된 순서대로 하드웨어를 검사해서 첫번째 섹터확인
 - 5) 첫번째 섹터의 부트코드를 CPU가 처리하여 윈도우 운영체제가 위치한 부팅 가능한 파티션의 위치를 알아냄

*디스크부트코드가 사라지면 BIOS 는 부팅가능 장치를 찾을 수 없어 오류 발생

2.3. 하드디스크 기술

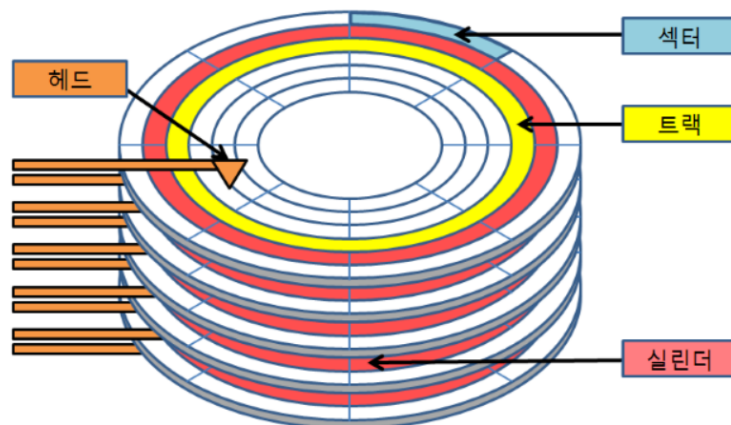


1. 하드 디스크 기초
2. 디스크 접근 방법
3. 쓰기 방지
4. 숨겨진 데이터 위치

하드디스크 위치정보와 내부구조

- 플래터 : 데이터가 쓰이는 곳
 - 위, 아래로 코팅이 되어있으며 플래터를 읽기위한 헤더가 위, 아래로 존재
- 플래터가 회전하고 헤더가 데이터를 읽는 구조

Structure Of Hard Disk



- 트랙 : 링 한개
 - 트랙 제일 바깥부터 0번
- 섹터 : 트랙을 동일한 크기 (512바이트) 로 나눈 한 부분
 - 하드디스크 주소가 할당되는 가장 작은 단위
 - 1번부터 시작

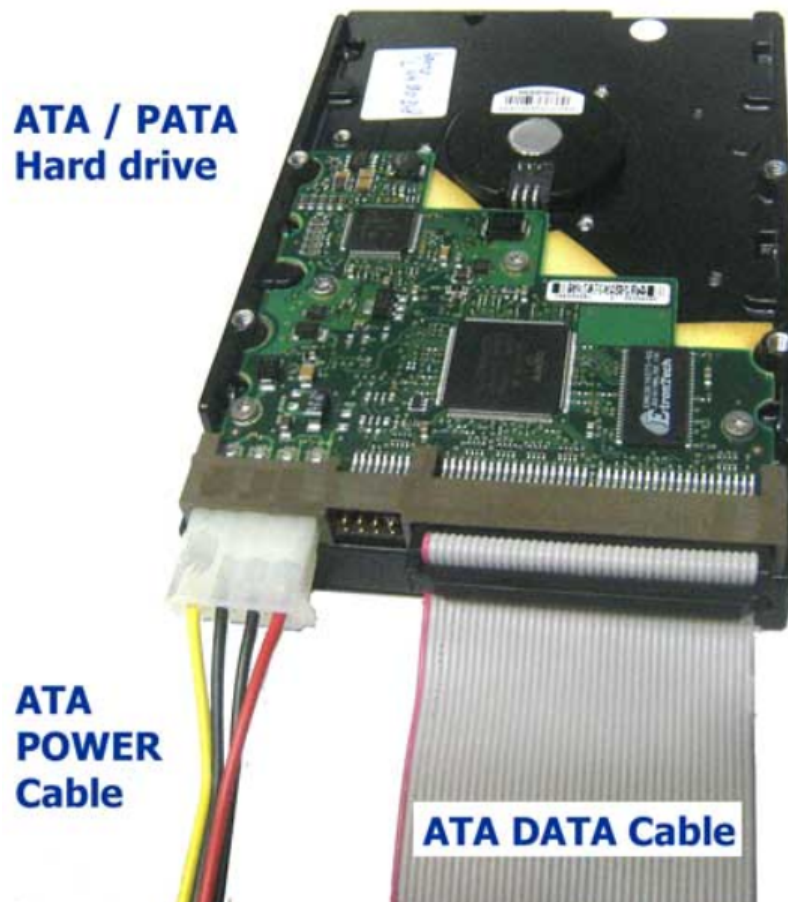
- 실린더 : 트랙의 주소 지정을 위해
 - 실린더 0 = 플래터 위, 아래의 트랙 0
- CHS : 특정 데이터가 들어있는 위치를 보기위해
 - C (실린더 주소), H(앞, 뒤 헤더 번호), S(섹터 주소) 를 이용

ATA/IDE *인터페이스

*인터페이스 : 서로 다른 두개의 시스템과 장치 사이에서 정보나 신호를 주고 받는 경우 접점이나 경계면

- 하드디스크에만 적용되는 인터페이스
- IDE(Integrated Disk Electronics) : 디스크 전자장치

→ IDE 디스크가 사용하는 인터페이스가 ATA



섹터주소유형

- 디스크에 데이터를 읽고 쓰기 위해 섹터에 주소를 지정하는 방식
 - 실제 주소 (물리주소) 이다
- CHS 방법
 - 실린더, 헤드, 섹터로 주소를 지정
 - ATA와 ATA와 통신을 위한 BIOS는 다음과 같이 비트를 할당함
 - ATA : 16비트(C), 4비트(H), 8비트(S)
 - BIOS : 10비트(C), 8비트(H), 6비트(S)
 - 서로 주소 표현을 위해 할당해주는 비트수가 달라서 최대 504MB만 인식
저장용량이 커진 지금은 사용하지 않음
- LBA 방법
 - 시작하는 숫자 (0부터) 로 논리적으로 위치를 알아냄
 - CHS 방식을 LBA 방식으로 계산 가능

인터페이스 표준

- 지역과 회사마다 다르게 쓰이던 인터페이스 용어 정리
 - ATA-1
 - ATA-3
 - ATA/ATAPI-4
 - ATA/ATAPI-6
 - ATA/ATAPI-7
 - SATA
- 밑으로 갈수록 최근

디스크 명령어

- 명령어는 어떤 방식으로 전달되는가
 - 1) 컨트롤러에는 래지스터가 존재한다 이 래지스터에 데이터를 적어줌
 - 2) 명령 래지스터에 명령을 작성하면
 - 3) 이때 명령 래지스터가 하드디스크로 처리하드디스크는 명령래지스터에 쓰기 전까지 아무것도 못함

하드디스크 비밀번호

- 디스크 읽을 시 비밀번호 해제 필수
- ATA-3 에서 BIOS 나 소프트웨어가 설정할 수 있는 보안 기능 소개
- 하드디스크 비밀번호 종류
 - 마스터 비밀번호
 - 보안업체, 회사에서 설정한 비밀번호
 - 사용자 비밀번호
 - 일반 사용자들이 하드디스크를 보호할 수 있는 방법
- 하드디스크 비밀번호 방식
 - High-Security : 사용자, 마스터 비밀번호로 모두 잠금해제 가능
 - Maximum-Security : 사용자 비밀번호로 잠금 해제 가능하고 마스터 비밀번호로 해제 시 내용이 모두 지워짐

HPA(Host Protected Area)

- 디스크를 저장할 수 있는 디스크의 특별한 영역
- 일반 사용자는 볼 수 없음
- ATA 명령으로 설정 가능
- 기본값은 0으로 설정된 경우가 많음
- HPA 확인 방법
 - READ_NATIVE_MAX_ADDRESS : 물리적 주소의 최대값을 반환
 - IDENTIFY_DEV_ICE : 사용자가 접근가능한 섹터 수를 반환
 - 위가 20GB 밑이 19GB 즉, $20-1 = 1GB$ 가 HPA
- HPA 공간 설정

- SET_MAX_ADDRESS

DCO (Device Configuration Overlay)

- HPA 와 마찬가지로 데이터 숨김영역
- **DEVICE_CONFIGURATION_IDENTIFY** 명령으로 전체 기능과 크기 보기 가능
- **DEVICE_CONFIGURATION_SET** : DCO 만들기
- **DEVICE_CONFIGURATION_RESET** : DCO 삭제

직렬 ATA

- 기존 병렬로 이루어지던 케이블의 속도 문제와 유연성을 해결하기 위하여 ATA-7에서 직렬 ATA 개발
- 더 많은 비트 전송

BIOS와 직접접근

*어떻게 소프트웨어가 하드디스크에 연결?

- 컨트롤러의 BIOS 접근
 - 소프트웨어와 하드 디스크 컨트롤러 사이의 통신필요
- 컨트롤러의 BIOS 접근
 - 직접접근은 빠르지만 소프트웨어가 하드웨어에 알아야할 부분들이 많이 존재













BIOS

- 컴퓨터 시작시 사용
- 현재 설치된 디스크들의 세부사항 결정
- 운영체제나 소프트웨어 서비스를 제공할 때 사용되는 인터럽트 테이블 적재

SCSI 드라이브

- Small Computer System Interface 약자
 - SCSI 타입과 ATA의 차이
 - ATA는 오직 40, 44핀
 - SCSI 는 다양한 모양과 종류가 있음
 - 가장 큰 차이점은 SCSI는 컨트롤러가 없음
 - SCSI는 ATA디스크 처럼 크기제한이 없다
- 항상 32비트와 64비트 LBA 주소를 사용하기 때문

커넥터 타입

External SCSI Connectors		
		
Centronics 50-Pin	DB 25-Pin	DB 50-Pin
		
High Density DB 50-Pin (Clip Type)	High Density DB 50-Pin (Screw Type)	VHDCI 68-Pin
		
High Density DB 68-Pin (Clip Type)	High Density DB 68-Pin (Screw Type)	High Density Centronics 50-Pin
		
High Density Centronics 60-Pin	High Density Centronics 68-Pin	HDI30 (Apple/Mac)

Chapter 3. 하드디스크 데이터 수집



하드디스크 저장장치의 데이터 분석을 다룸

하드디스크 데이터 수집 방법 이론 (시스템 보전 단계)

리눅스 dd 도구를 이용한 실습

3.1. 소개



하드디스크 복사본 만들기

수집 절차

- 수집 : 원본 디스크 저장 장치에 있는 데이터를 복사해서 복사본을 만드는 과정
- 512 바이트부터(최소 수집단위) 바이트 단위로 복사가 이루어짐
- 데이터 묶음을 Chunk 라고 부름

데이터 수집 계층

*챕터 1 데이터 계층 참고 / 데이터 수집은 어떤 계층 단위로?

- 각 계층(5가지) 에서 하는 데이터 수집은 데이터 손실 우려로 증거가 있는 가장 아래 디스크 계층에서 데이터를 수집해야함
- 디스크 수준 데이터 수집 → 대부분의 경우
 - 각 파티션 내의 모든 섹터의 복사본을 얻게됨
 - 지워진 파일 복구 가능
 - **파티션에 할당되지 않은 섹터들은 복사가 불가능**
- 백업 유틸리티 사용
 - 파일을 기준으로 복사
 - 지워진 파일은 복구 불가능
 - 파일시스템과 숨겨진 데이터를 찾을 수 없음
 - 백업만 가능한 데이터가 존재할 때 백업 유틸리티를 이용해서 수집
 - 누가 시스템에 접근했는지 증거를 제공
- IDS(침입탐지시스템) 에는 공격과 관련된 로그가 존재

- 이때 IDS가 해킹당하지 않았다면 로그를 이용한 분석을 위해서는 백업 유틸리티 이용
- IDS도 해킹 당했다면 모든 데이터의 분석 (디스크 수준) 이 필요

3.2. 원본 데이터 읽기



1. 디스크 읽는 방법
2. 복사본에 쓰는 것과 관련된 사항
3. 데이터 접근 방법
4. 오류 처리방법
5. 분석 대상 드라이브에 데이터를 쓰지 않는 방법

직접접근과 BIOS 접근

- BIOS 접근이 상대적으로 쉬워보이나 실제 수집에서는 직접접근
 - BIOS의 부정확성 때문에

정적수집과 동적수집

- 정적 수집 : 운영체제가 정적상태에서 도움 없이 복사
- 동적 수집 : 운영체제가 동적상태에서 도움으로 복사
 - 공격자가 운영체제를 변조했을 가능성
 - 수집하는 동안 잘못된 데이터를 제공하도록 하는 악성 소프트웨어가 존재할 수 있음
- 수집 시 동적 수집은 가급적 피한다
- 도스 플로피 또는 리눅스 CD를 이용해 부팅
 - 플로피나 CD는 특정 데이터를 수정하거나 드라이브를 마운트 할 수 없도록 설정되어 있기 때문에

오류 처리

- 드라이브 전체가 오류
- 몇개의 섹터가 손상
- 불량 섹터의 주소에 0을 씌

숨겨진 영역 또한 수집

- HPA → 대다수 분석 도구가 이 영역을 탐지
- DCO → 제거시에는 문서화 작업 필요

하드웨어 쓰기방지 장치

*명령자체를 차단하는 방식




- 수집 과정자체는 원본 파일을 건드리는 것이므로 원본이 변경될 가능성이 있다
- 하드웨어 쓰기방지 장치는 컴퓨터와 저장장치 사이에 놓는 장치

- 쓰기 방지 장치는 ATA, SCSI, Firewire, USB, Serial ATA 같은 많은 저장 인터페이스를 지원함

소프트웨어 쓰기 방지 장치

Software Write Block

Software Write Block Specs

 <https://www.nist.gov/itl/ssd/software-quality-group/computer-forensics-tool-testing-program-cftt/cftt-technical/software>

- 대부분의 디지털 포렌식 도구들은 도스기반
- 디스크에 접근하기 위해 INT13h 사용
- BIOS 서비스 코드가 위치한 인터럽트 테이블을 수정해서 동작
 - 인터럽트 테이블
 - BIOS가 제공하는 모든 서비스의 엔트리 존재
 - 각 엔트리는 서비스 코드의 위치를 알려주는 주소를 포함
- 인터럽트 테이블을 수정하여 작동

→ 소프트웨어는 BIOS 를 우회해서 쓰기 데이터를 컨트롤러에 쓸 수 있고

BIOS는 컨트롤러에 직접 접근해서 디스크에 데이터를 쓰기 때문에

하드웨어쓰기방지장치만큼 효율적이지 않음

3.3. 출력 데이터 쓰기



원본디스크에서 데이터를 읽은 후 보관이 필요

목적지

- 데이터를 저장하는 방식
- 방법 2가지

- 직접 : CD나 디스크에 복사와 복제
 - 원본 섹터 위치와 복사본 섹터 위치가 일치
 - 복사 전에 디스크를 0으로 초기화 하지 않으면 위치에 혼란이 생김
- 파일 : 파일로 복사 → 이미징
 - 위치가 실제 이미지와 다르기 때문에 자동으로 마운트 불가

이미지 파일 형식

*파일로 저장되는 이미지 형식

- 미가공 이미지
 - 원본 장치 데이터 만을 포함하기 때문에 원본과 비교가 쉬움
- 내장 이미지
 - 원본 장치에 데이터를 포함하고 추가로 해시, 날짜, 시간 같은 부수적인 데이터 포함

이미지 파일 압축

- 이미지 생성시 압축옵션이 존재하면 저장공간을 늘릴 수 있다
- 압축된 이미지 분석시 압축해제 필요
 - 장점 : 이미지 파일을 작은크기의 형태로 수집 가능
 - 단점 : 압축 형식을 지원하는 도구 수 한정

네트워크 기반 수집

- 네트워크를 이용해 원격의 컴퓨터에 이미지 파일 생성 가능
- 이 경우 데이터를 원본 디스크에서 읽어 네트워크를 통해 목적지 호스트에 전송한 후 파일에 쓴다
- 분석 대상 디스크에 접근이 불가능 하거나, 올바른 데이터가 없거나, 분석 대상 디스크를 위한 인터페이스를 얻을 수 없는 경우 사용

무결성해시

- 일부 수집 도구들은 수집시 해시를 계산하고 일부는 별도의 장치가 필요함
- 내장 데이터, 미가공+외부(메타데이터) 형식으로 저장
- 해시 자체가 보안 기능을 제공하는 것이라기보다 값으로 비교하며 무결성을 확인하는 것
- 원본과 이미지의 해시도구는 다른것을 사용하는 것이 좋음

3.4. dd를 이용한 실습

- dd 도구
 - 명령줄 기반
 - 한 파일에서 데이터 묶음을 복사해서 다른 곳에 쓰기를 함으로써 수집
 - 입력되는 데이터의 타입이 어떤지 신경쓰지 않고 오로지 파일의 형태로만 인식
- if= : 읽을 디스크 이미지를 넣는 플래그
- of=: 저장될 출력파일
- bs=: byte size로서 블록의 크기
- skip : 읽기전에 건너뛴 블록개수(블록크기는 bs에 따름)
- count : 입력에서 출력으로 복사할 블록의 개수

EX) 원본의 크기가 1024바이트인 데이터(file1.dat)를 512byte 단위로 복사한다고 가정

#dd if=file1.dat of=file2.dat bs=512

2+0 records in

2+0 records out

2+0에서 2는 두($1024/512=2$) 개의 블록을 의미한다. 뒤에 0은 다채우지 못한 블록을 의미

EX2) 원본의 크기가 1500인 블록을 똑같이 옮긴 결과

2+1 records in

2+1 records out

입력소스

- 리눅스는 각 저장장치와 파티션을 위한 장치가 따로 존재
- 윈도우는 디스크를 참조할 수 있는 '\\'를 사용해야 한다.
- 기본 블록 크기는 512byte 인데 'bs='를 통해 최대 1GB바이트까지 설정이 가능하다. (2kb~8kb 추천)
- 리눅스는 BIOS를 거치지않고 바로 하드웨어에 접근하기 때문에 쓰기방지를 위해서는 **하드웨어 쓰기 방지를 사용해야 한다**

HPA

- 파일단위복사는 HPA 영역을 알아낼 수 없음
- dd도구 역시 파일단위로 수집
- HPA영역을 알아내기 위해서는 다양한 방법이 존재한다.
 - 1) dmseg로그로 존재확인하기
 - 크기제한에 걸리거나 다른 응용프로그램이 로그를 덮어버리는 문제가 발생가능
 - 2) Hdparm 도구 사용
 - 하드디스크의 세부사항을 볼 수 있음
 - hdparm -i 를 이용하면 섹터 전체수를 알아낼 수 있다
 - 이 값과 제조사에서 제공하는 섹터수를 비교하면 HPA영역을 알아낼 수 있다
 - 3) The Sleuth kit의 diskstat 도구 사용
 - 최대 원본 주소(Maximum native address)와 최대 사용자 주소를 알아낼 수있다 (이 둘의 차이가 HPA영역)

위의 방법들로 HPA영역의 존재를 알아내면 'setmax'와 같은 도구로 최대 주소를 재설정할 수 있다.

(setmax --max XXXXXXXXX /dev/hdb)

※ 하지만 이렇게 재설정하기 전 HPA 시작위치를 기록해두고 다른 디스크로 미리 테스트 필요

출력 목적지

- dd의 출력은 새로운 파일이나 새로운 저장장치 중 하나이다.

두가지 출력방법

여기서 of=를 지정하지 않으면 화면에 출력한다. -> 해시값계산, ASCII문자열 추출, 네트워크전송에 유용

ex1) **#dd if=/dev/hda bs=2k | md5sum**

- > 화면에 MD5 해시값을 보여준다.

ex2) netcat, cryptcat을 이용해서 원격에 있는 서버로 전송

nc -l -p 7000 > disk.dd //7000번 포트로 들어오는 값 disk.dd에 저장

dd if=/dev/hda bs=2k | nc -w 3 10.0.0.1 7000 //목적 아이피의 7000번 포트로 출력(전송)

오류처리

- dd가 파일을 읽는 동안 오류가 발생하면 기본적으로는 복사를 멈춤
 - **'conv=noerror'** 플래그 설정시 dd는 오류보고를 하고 하던 일을 계속 진행
- 불량 데이터 블록을 건너뛰다 (잘못된 크기, 잘못된 주소를 가질 수 있다)
- 이미지 주소를 관리하기 위해서는 **sync**플래그를 사용
- 블록크기의 데이터 묶음을 쓰도록 하고 다 못채울 경우 나머지를 0으로 표시한다. 또한 오류도 0으로 처리

#dd if=/dev/hda of=hda.dd bs=2k conv=noerror, sync

암호해시

*보통 dd를 사용하면서 해시를 얻으려면 'md5sum'과 같은 명령어를 추가로 사용해야했음

- 복사되는 데이터의 해시를 계산하는 dd버전이 개발 됨

1. **dcfldd** : MD5 가능

2. **ddcidd** : MD5, SHA-1, SHA-256 등 같이 계산할 수 있다

dcfldd if=/dev/hda of=/mnt/hda.dd bs=2k hashwindow=1M hashlog=/mnt/hda.hashes

(hashwindow= 해시 계산 크기(여기서는 매 1M바이트마다 해시 계산))