Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.29, NO.1, Feb. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.1.77

리눅스 커널에 따른 메타데이터 기반 파일 복원 연구*

신 영 훈,[1†] 조 우 연,[1] 손 태 식[1,2‡]
[1]아주대학교 컴퓨터공학과, [2]아주대학교 사이버보안학과

Study on File Recovery Based on Metadata Accoring to Linux Kernel*

Yeonghun Shin,[1†] Woo-yeon Jo,[1] Taeshik Shon[1,2‡]
[1]Department of Computer Engineeering, Ajou University
[2]Department of Cyber Security, Ajou University

요 약

최근의 리눅스 운영체제는 자동차의 콘솔, CCTV, IoT기기, 모바일기기 등에 이르기까지 사용량이 증가하여 커널 버전 또한 다양하게 활용되고 있다. 해당 기기들은 범죄 수사 시 강력한 증거자료로 활용될 수 있기 때문에 파일 삭제를 통한 증거 인멸의 우려가 있다. Ext 파일시스템 포렌식은 기기의 종류에 의존하지 않고 삭제된 파일을 복원할 수 있기 때문에 심도 있는 연구가 수행되어 왔다. 하지만 커널에 따라 달라질 수 있는 파일시스템의 특성은 고려되지 않은 채 연구를 진행해왔다. 이러한 문제점은 실제 수사에서 상이한 버전의 커널을 대상으로 분석을 시도할 경우 파일시스템의 주요 메타데이터 분석이 제대로 이루어지지 않아 수사능력을 저해하고 증거능력의 의심을 받는 등 심각한 상황을 초래할 수 있다. 실제 수사 현장에서는 다양한 배포판 및 커널 버전의 리눅스 파일시스템을 대상으로 수사가 진행될 수 있기 때문에, 실증적으로 활용될 수 있도록 리눅스 배포판 및 커널 버전별로 파일 삭제 시 발생하는 메타데이터 변화 차이에 대한 분석이 필요하다. 따라서 본 논문은 이에 대한 해결 방안으로 리눅스 커널에 따른 메타데이터의 차이를 분석하고 삭제된 파일의 복원을 수행한다. 이후 수사기관은 Ext 파일시스템 포렌식 수행 시 리눅스 커널 버전 차이에 의해 생기는 메타데이터 변화에 대한 고려가 필요하다.

ABSTRACT

Recent Linux operating systems having been increasingly used, ranging from automotive consoles, CCTV, IoT devices, and mobile devices to various versions of the kernel. Because these devices can be used as strong evidence in criminal investigations, there is a risk of destroying evidence through file deletion. Ext filesystem forensics has been studied in depth because it can recovery deleted files without depending on the kind of device. However, studies have been carried out without consideration of characteristics of file system which may vary depending on the kernel. This problem can lead to serious situations, such as those that can impair investigative ability and cause doubt of evidence ability, when an actual investigation attempts to analyze a different version of the kernel. Because investigations can be performed on various distribution and kernel versions of Linux file systems at the actual investigation site, analysis of the metadata changes that occur when files are deleted by Linux distribution and kernel versions is required. Therefore, in this paper, we analyze the difference of metadata according to the Linux kernel as a solution to this and recovery deleted file. After that, the investigating agency needs to consider the metadata change caused by the difference of Linux kernel version when performing Ext filesystem forensics.

Keywords: Digital Forensics, Filesystem, Linux, Ext, File Recovery

Received(10. 11. 2018), Modified(11. 14. 2018), Accepted(11. 26. 2018)

* 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2018-2016-0-00304)

* 이 논문은 2018년도 정보(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2018-0-01000, 디지털 포렌식 통합 플랫폼 개발)

† 주저자, syh2347@ajou.ac.kr

‡ 교신저자, tsshon@ajou.ac.kr(Corresponding author)

I. 서 론

리눅스는 초기 미닉스와 파일시스템을 공유해서 사용했으나 파일 이름 길이의 제한과 최대 크기가 작아 1992년 리눅스 커널 0.96c 버전에서부터 리눅스 전용 파일 시스템인 Ext 파일시스템을 개발하여 채용하였다. 이후 1993년 개발된 EXT2 에서 EXT4 까지 계속해서 사용되어오고 있다. 현재 리눅스를 기반으로 동작하는 시스템에는 서버에서부터 CCTV, 차량용 블랙박스, 모바일 디바이스, 클라우드 서버 등 다양한 기기가 존재한다. 리눅스의 오픈된 성향은 현재 개발되는 대부분의 기기에 적용될 수 있는 원동력이 되어 매우 급진적인 사용량 증가를 가져오고 있다. 이러한 추세의 영향으로 많은 모바일 기기 대상 디지털 포렌식 연구들이 수행되고 있다. 모바일 디바이스의 경우 안드로이드 기반 스마트폰이 리눅스 커널을 기반으로 동작하기 때문에 스마트폰 사용량의 증가에 따라 리눅스 커널의 기본 파일시스템인 Ext 파일시스템의 사용량이 증가하게 되었다. 안드로이드 운영체제를 기반으로 동작하는 모바일 디바이스는 최근 빠른 발전으로 점유율이 90% 이상으로 집계되며[1], 사용자의 위치정보, 검색기록 등 일상의 정보를 갖고 있으므로 범죄 발생 시 강력한 증거가 된다[2]. 하지만 범죄자가 기기 내부에 저장된 사건과 관련된 데이터를 의도적으로 삭제하는 경우가 많기 때문에 Ext 파일시스템 포렌식의 중요성과 필요성은 점차 증가하고 있다[3][4].

데이터베이스 서버의 경우에도 위 모바일 디바이스의 경우와 비슷한 양상을 보이고 있다. 다수의 DBMS(Database Management System)들은 SQL계열과 NoSQL계열을 가리지 않고 대부분 리눅스를 지원한다. 또한 안정성을 중요시 여기는 서버의 경우 오랜 기간 사용되어 오래된 리눅스 배포판 버전이 그대로 탑재된 경우를 자주 접할 수 있다. 따라서 오래된 리눅스 배포판/커널 버전에 따라 달라질 수 있는 파일시스템 영역을 검증할 수 있는 디지털 포렌식 관점의 연구가 수행되어야 한다.

근래 활발히 연구 개발되고 있는 자동차 산업의 경우에도 Ext 파일시스템 포렌식이 요구된다[12]. 리눅스 시스템이 채용된 예시로 차량용 네트워크 표준으로 사용되고 있는 임베디드 네트워크 CAN(Controller Area Network)가 있으며, 토요타는 차량 인포테인먼트 시스템에 AGL(Automotive Grade Linux, 리눅스 재단이 제공하는 자동차용 오픈소스 리눅스 프로젝트)을 사용한다. 이처럼 리눅스 파일시스템은 자율 주행 차량, 전기 자동차 등 다양한 분야의 자동차 산업에서도 활용되고 있어 디지털 포렌식 연구의 필요성을 확인할 수 있다.

Ext 파일시스템은 높은 사용률뿐만 아니라 파일시스템을 이루는 주요 메타데이터의 구조가 공개되어 있어 다양한 디지털 포렌식 연구가 활발히 수행되어 왔으며, 파일 복구에 관한 연구 또한 존재한다[5]. 대표적인 복구 기법으로 파일 카빙 기법과 메타데이터를 이용한 복구 기법 두 가지로 분류할 수 있으며, 두 기법 모두 장단점이 존재한다. 메타데이터를 이용한 복구 기법은 Ext3/4 파일시스템에서 저널링 기법을 활용하여 낮은 오탐률과 높은 복구율을 확보할 수 있다[1]. Ext2 파일시스템은 삭제 이후 남아있는 메타데이터를 통해 파일을 복구할 수 있지만 메타데이터가 지워진 파일시스템의 경우 복구가 불가능해진다.

현재 Ext4가 널리 쓰이고 있지만 여전히 포렌식의 대상이 되는 서버, CCTV 등 다양한 기기에 Ext2/3가 쓰이고 있다. Ext4의 경우 Ext3와 같은 저널링 기법을 사용하기 때문에 본 논문에서는 주로 Ext2/3을 사용하는 파일시스템의 차이점 분석을 통해 실제 포렌식 수사 과정에 활용하는 방안을 모색한다. 결과적으로 다양한 리눅스 배포판이나 커널을 가진 시스템들을 다룰 때 발생할 수 있는 분석 결과의 차이 등을 고려할 수 있어 실제 현장 포렌식 수사관들에게 실질적인 도움이 될 수 있는 방안을 제시하고자 한다.

본 논문에서는 Ext2/3 파일시스템에서 커널 버전에 따라 파일 삭제 시 발생하는 메타데이터의 변화가 파일 복구에 미치는 영향과 리눅스 배포판 종류에 따라 파일 삭제 시 발생하는 메타데이터 변화의 차이가 있는지에 대해 실험을 통해 분석·검증한다.

2장과 3장에서는 Ext2/3 파일시스템의 구조, 파일 삭제 시 발생하는 메타데이터의 변화 그리고 파일 카빙을 통한 복구 방법과 메타데이터 분석을 통한 파일 복구 방법의 장단점에 대해서 다룬다. 4장에서는 앞서 다룬 Ext 파일시스템의 메타데이터 구조를 이용하여 직접 파일을 삭제할 때 발생하는 메타데이터의 변화와 커널 버전 별 메타데이터의 차이, 발생한 메타데이터 차이가 파일 복구에 미치는 영향을 실험을 통해 보인다. 마지막으로 5장에서는 본 논문의 결과와 향후 연구에 대해 언급한다.

II. 관련연구

리눅스 사용량이 늘어나면서 Ext 파일시스템의 구조에 대한 연구가 많이 진행되었다. 저널링 기법을 사용하지 않는 Ext2/3 파일시스템의 구조는 대부분 파악되어 있으며[5], 저널링 기법을 사용하는 Ext4 파일시스템의 구조에 대한 연구 또한 많이 진행되고 있다[7].

위에서 언급한 파일 복구를 위한 두 가지 기법 중 파일 카빙 기법은 메타데이터 정보가 지워지더라도 복구를 할 수 있다는 장점이 있다[10][11]. 파일 카빙 기법은 파일 자체 데이터의 구조를 분석하여 파일시스템의 메타데이터와 무관하게 파일의 파편을 모아 하나의 파일을 복구한다. 하지만 이러한 방식은 파일의 크기가 커 단편화가 많이 되어있는 파일일수록 복구하기 어렵다는 단점이 있으며[2]. 오탐(False Positive)이 발생할 가능성이 있기 때문에 복구된 파일에 대한 신뢰성을 보장할 수 없다[8].

메타데이터 분석을 통한 파일 복구 기법의 경우 파일 복구에 관련된 메타데이터 정보가 남아있다면 파일의 복구가 가능하며, Ext3/4 파일시스템의 경우는 일반적인 메타데이터 정보가 지워지더라도 저널 영역이라 불리우는 파일시스템의 백업 영역이 남아있다면 메타데이터의 복구가 가능하다. 또한, 파일의 완전 복구가 되지 않더라도 복구된 부분까지는 신뢰성을 보장할 수 있다는 장점이 있다. 하지만 이러한 방법은 파일시스템마다 서로 다른 메타데이터 동작 구조를 완전히 파악해야 하며, 파일시스템의 동작이 일부라도 변경된 경우 다시 분석 및 개발을 수행해야 한다는 단점이 존재한다.

포렌식 수사를 통해 복구된 자료는 법적 증거로써 채택되기 위해서는 무결성/신뢰성이 보장되어야 한다. 메타데이터 분석을 통한 파일 복구 기법이 이를 보장할 수 있으며, Ext3/4 파일시스템의 특징 중 하나인 저널 영역에 대한 연구가 진행됨에 따라[1] 메타데이터 정보가 지워져도 저널에 있는 Inode의 백업을 통해 파일을 복구하는 기법에 대한 연구 또한 많이 진행되었다[6]. 최근에는 파일 카빙 기법과 메타데이터 분석 기법을 함께 사용하는 아이노드 카빙 기법에 대한 연구[9], 메타데이터를 이용해 파일을 복구하는 기법을 위해 파일 삭제 시 발생하는 메타데이터의 변화를 파악하는[7] 등 메타데이터의 가치에 관한 연구가 진행되었다[8].

현재 대부분의 연구는 Ext 파일시스템에서의 메타데이터 분석이나 파일 카빙에만 초점을 맞추고 있다. 따라서 실제 현장에서 신뢰성 확보를 위해서 수반되어야 하는 연구인 리눅스 배포판이나 커널 차이에 의해 생기는 차이점 등에 대한 연구가 미미하다. 따라서 본 논문에서는 Ext 파일시스템에 대해 디지털 포렌식 연구들이 수행되기 이전에 필수적으로 수행되었어야 하는 연구로, 리눅스 배포판 및 커널 버전별 파일 삭제 시 생기는 메타데이터의 변화에 차이가 존재하는지 비교를 통해 분석하여 발생한 차이가 파일 복구에 미치는 영향을 분석한다.

III. 리눅스 커널 및 배포판별 분석을 위한 메타데이터 구조

본 장에서는 배포판/커널별 비교 분석의 대상이 되는 Ext2/3 파일시스템의 메타데이터 정보에 대해 먼저 살펴본다. Ext 파일시스템은 부팅을 위한 부트로더가 기록된 MBR 영역 이후 다수의 Block Group으로 구성되어있다. Fig.1.은 Ext 파일시스템의 구조이다. Block Group 0의 경우 padding을 위해 맨 앞 1024bytes를 사용한다. 따라서 파일시스템의 0x400번지에서 SuperBlock을 발견할 수 있고, Group 0 이외에는 Padding 없이 SuperBlock 공간으로 시작된다.

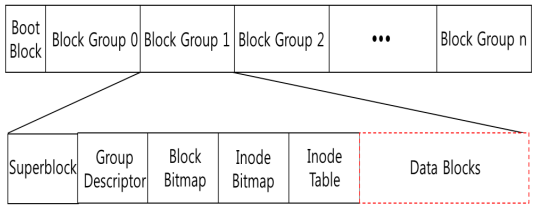

Fig. 1. Ext File system Layout

3.1 SuperBlock

SuperBlock은 파일시스템을 이루는 필수적인 메타데이터 정보인 데이터 블록의 크기와 개수, 저널 사용 여부, 저널의 위치 등을 포함하는 블록이다. 따라서 Ext 파일시스템 분석은 SuperBlock에서 시작해야하며, 파일시스템 시작점부터 고정된 위치 0x400(1024bytes, Offset)에 위치하기 때문에 쉽게 찾을 수 있다. Superblock은 기본적으로 하나의 블록을 차지하나 고정된 Offset을 갖기 때문에 블록의 크기에 따라 Fig.2.와 같이 첫 번째 블록에 올

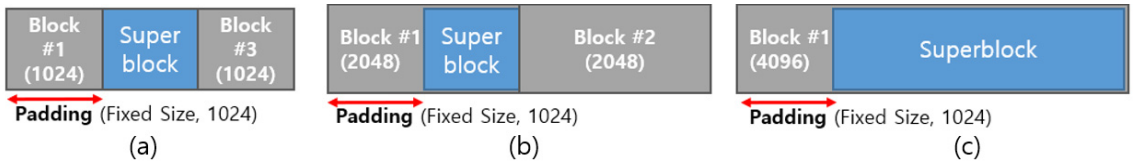

Fig. 2. (a) - If the block size is 0x400 (1024), the Superblock located in the second block
(b) - If the block size is 0x800 (2048), the Superblock located in the first block
(c) - If the block size is 0x1000 (4096), the Superblock located in the first block

수도 두 번째 블록에 올 수도 있다. SuperBlock 공간에서 삭제된 파일을 복원하는데 관여하는 주요 정보는 Table 1.과 같다. Table 1.의 Value 값은 본 논문에서 실험을 진행한 Ubuntu14.04의 Ext2 100MB 이미지에서 얻은 값이다. s_magic 필드는 고정 값으로 모든 이미지에서 동일하게 0xEF53으로 기록되며 s_feature_compat 필드의 경우 저널을 가질 때 0x0004의 고정된 값을 가진다. s_feature_incompat 필드의 경우 Ext4일 경우 0x0040의 고정된 값을 가진다. 위의 세 필드를 이용해 해당 Ext 파일시스템의 버전을 알아낼 수 있다. 또한, 위 세 필드를 제외한 나머지 필드는 가변적인 값을 가지며 s_blocks_count_lo 필드와 s_blocks_per_group 필드를 이용해 전체 Block Group의 수를 계산할 수 있다.

Table 1. Structure of SuperBlock

Offset	Name	Value	Descriptor
0x00 ~ 0x03	s_Inodes_count	0x2865	Total inode count
0x04 ~ 0x07	s_blocks_count_lo	0x19400	Total block count
0x10 ~ 0x13	s_free_Inodes_count	0x6504	Free inode count
0x18 ~ 0x1B	s_log_block_size	0x00 (2^{10})	Block size (2^{10+X})
0x20 ~ 0x23	s_blocks_per_group	0x2000	Blocks per group
0x28 ~ 0x2B	s_Inodes_per_group	0x07C8	Inodes per group
0x38 ~ 0x39	s_magic	0xEF53 (fixed)	Magic signature
0x58 ~ 0x59	s_Inode_size	0x80	Size of inode structure
0x5C ~ 0x5F	s_feature_compat	0x38	Compatible feature set flags
0x60 ~ 0x63	s_feature_incompat	0x02	Incompatible feature set flags

3.2 Group Descriptor

Group Descriptor는 파일시스템의 Blcok Group 각각에 대한 정보를 갖고 있으며, 기본적으로 32bytes로 구성되어있다. Group Descriptors도 SuperBlock과 동일하게 각 Block Group에 백업이 존재하며, 32bytes(Default) 단위로 Block Group의 수만큼 이어서 모여 있다. Group Descriptor 영역에서 삭제된 파일을 복원하는 입장에서 중요한 정보는 Table 2.와 같다. Table 2.의 Value 값은 본 논문에서 실험을 진행한 Ubuntu14.04의 Ext2 100MB 이미지에서 얻은 값이다. Group Descriptor의 필드들을 통해 Block Bitmap, Inode Bitmap, Inode Table의 위치를 찾아낼 수 있다.

Table 2. Group Descriptor Structure

Offset	Name	Value	Descriptor
0x00 ~ 0x03	bg_block_bitmap_lo	0x0103	Lower 32bits of location of block bitmap
0x04 ~ 0x07	bg_Inode_bitmap_lo	0x0104	Lower 32bits of location of inode bitmap
0x08 ~ 0x0B	bg_Inode_table_lo	0x0105	Lower 32bits of location of inode table

3.3 Inode Bitmap/Table

Inode는 각각의 파일이나 디렉토리에 대한 정보를 담고 있다. 실제 파일이 쓰여진 데이터 블록의 위치 정보를 포함하고 있는 파일 복구에 필수적인 정보이며, 한 Inode의 크기는 SuperBlock의 s_inode_size를 통해 알 수 있다. Inode Bitmap을 통해 Inode가 사용되고 있는지 아닌지를 알 수 있고, Inode Table에서는 사용되고 있는 Inode에 대한 정보를 알아낼 수 있다. 이러한 점을 이용해 Inode Bitmap과 Inode Table을 비교하여 파일이 삭제되었는지도 확인할 수 있다.

3.3.1 Inode Bitmap

Inode Bitmap은 현재 Inode들의 할당 상태를 나타낸다. Inode Bitmap의 정보가 0이면 미할당, 1이면 할당된 상태이다. Inode Bitmap은 비트 단위로 읽는 순서가 중요하다. 한 바이트씩 순차적으로 읽되, 각 바이트를 비트열로 바꾸면 LSB부터 순서대로 읽어야 한다. Fig.3.의 예시로 보면 1~11번 Inode까지는 할당된 상태이며, 12~16번 Inode는 미할당 상태임을 알 수 있다. 만약 Inode Bitmap에 0으로 할당되어 있는데 Inode Table에 Inode 정보가 존재한다면 삭제된 파일로 생각할 수 있다.

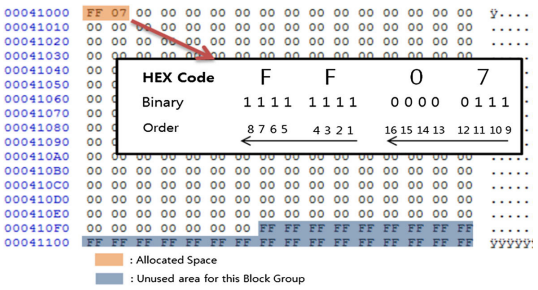

Fig. 3. The order to read inode bitmap[5]

3.3.2 Inode Table

Inode Table은 Inode가 연속적으로 저장되어 있는 구조를 말한다. Inode는 Table의 시작 위치부터 1번, 2번, 3번… 순으로 저장된다. Ext 파일시스템의 Inode Table은 1번 Inode부터 10번 Inode까지 시스템에서 예약되어 있고, 사용자의 데이터는 11번부터 순서대로 저장된다. 대표적으로 2번 Inode는 'Root 디렉토리'의 정보를 저장하고, 8번 Inode는 '저널 영역'(Ext3/4 한정)을 위해 예약되어 있다.

Table 3.은 Inode Table에 존재하는 한 개의 Inode를 이루는 필드들에 대한 정보이다. 아래의 표에서 알 수 있듯이 한 Inode는 한 파일에 대한 모든 정보를 갖고 있다. 따라서 하나의 파일을 복구하는데 있어서 Inode는 중요한 역할을 한다. 이 정보들을 이용하여 파일의 흩어져 저장되어 있는 실제 데이터들의 위치를 알아낼 수 있고, 그 데이터들을 합쳐서 파일을 복구해낼 수 있다[5]. 따라서, Inode Table의 유/무는 파일 복구의 성공과 실패에 많은 영향을 끼치며, Ext3/4 파일시스템에서는 저널링 기법을 이용해 Inode Table의 백업을 따로 저장해 둔다. Ext3/4 파일시스템에서는 Inode Table의 데이터가 손상되더라도 저널 영역에 있는 백업을 통해 파일을 복구할 수 있는 가능성이 있지만, Ext2 파일시스템에서는 Inode Table의 데이터가 손상된다면 파일 복구율이 현저히 떨어지게 된다.

Table 3. Inode Table Structure

Offset	Name	Descriptor
0x04 ~ 0x07	Size	Lower 32bit of size in bytes
0x14 ~ 0x17	Deleted Time	Deletion Time
0x1C~ 0x1F	Block Count	Lower 32bits of block count
0x28 ~ 0x57	Direct Pointers	Pointer to block of file data or pointers
0x58 ~ 0x5B	Single Indirect Pointer	
0x5C ~ 0x5F	Double Indirect Pointer	
0x60 ~ 0x63	Triple Indirect Pointer	

3.4 Journal

Ext3/4 파일시스템에서 사용되는 저널(Journal)은 시스템 크래시에 대비하여 메타데이터의 백업을

수행하기 위해 파일시스템의 메타데이터 변화를 기록하는 영역이다. 저널은 Table 4.와 같이 Journal, Ordered, Writeback의 세 가지 모드가 있으며, 파일시스템 생성 시 기본으로 설정되는 모드는 Ordered 모드이다.

Ordered 모드에서 저널은 Journal SuperBlock과 트랜잭션(Descriptor Block, Data Block, Commit Block)으로 구성된다. SuperBlock은 저널의 크기, 저널의 첫 블록 등 기본 메타데이터를 저장한다. Descriptor Block은 하나의 트랜잭션 내에 백업된 블록들에 대한 설명을 저장한다. Data Block은 실제로 백업된 데이터 블록을 저장한다. Commit Block은 하나의 트랜잭션이 저널에 완벽히 기록되었다는 것을 나타낸다.

Ext3/4에서 삭제된 파일의 아이노드는 'Delete Time', 'Block Addressing' 등의 필드가 0으로 변경되므로 저널 영역 확인이 필수적이다. 위에서 설명한 Journal을 통한 데이터 복구 방안은 Fig.4.와 같다.

Table 4. Journal mode

mode	features
Journal	Instead of storing only metadata, all files are written to the journal area and stored in the real area.
Ordered (Default)	Only metadata is recorded. Before storing metadata in the journal, the data is stored in the real area.
Writeback	Only metadata is stored in the journal area.

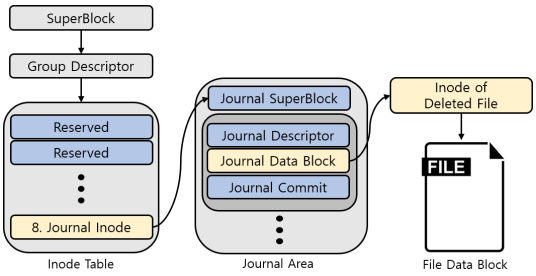

Fig. 4. Journal Area Recvoery Plan

IV. Ext2/3 파일시스템 메타데이터 비교 실험

본 장에서는 앞서 분석한 Ext 파일시스템에서 파일 삭제 시 메타데이터에 일어나는 변화를 확인한다. Ext2와 Ext3 파일시스템의 가장 큰 차이점은 저널링 기법의 유/무이기 때문에 이 점이 메타데이터 구조에 끼치는 영향과 파일 복구에 미치는 영향을 실험을 통해 분석한다. 또한 커널 버전별 차이점과 리눅스 배포판 간의 차이점이 있는지를 알아보기 위해 다양한 리눅스 배포판/커널 버전 중 몇 가지를 선정하며, 비교를 통해 얻은 차이점이 파일 복구에 미치는 영향에 대해 다룬다.

4.1 리눅스 배포판과 커널 선정

리눅스 배포판과 커널 버전 간의 파일 삭제 시 메타데이터의 차이점 유무를 알기 위해서 사용할 배포판과 커널을 몇 가지 선정했다. 선정 기준은 실제로 많이 사용되는 대중적인 리눅스 배포판이 수사현장에서 수사 대상으로 자주 사용될 것이라 생각됐기 때문에 5년 이상 높은 점유율을 유지하는 것으로 정하였다. Fig.5.은 2014년도까지의 리눅스 배포판별 점유율을 나타낸 그림이다. 위 그림뿐 아니라 DistroWatch Page Hit Ranking을 참고하면 최근 1개월까지 점유율을 알 수 있는데, Ubuntu와 Linux Mint, Debian, Redhat 등의 OS가 높은 점유율을 차지하고 있다. 본 실험에서는 위의 정보를 토대로 Linux Mint와 Ubuntu, 서버로 자주 사용되는 Redhat을 설치하였다. 각 OS는 다른 버전을 비교하여 파일 삭제 시 커널 버전에 따른 메타데이터의 차이를 분석하기 위해 최근의 버전 1개와 이전 버전 1개를 다운로드 받았으며, Redhat의 경우는 같은 버전의 Desktop용과 Server용 배포판을 사용하여 메타데이터의 차이를 분석하였다. 커널 버전

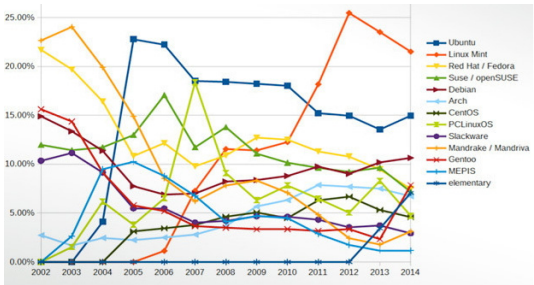

Fig. 5. Share based on Linux distribution

은 Ext4 파일시스템이 도입되지 않은 오래된 2.x 버전부터 최근의 4.x 버전까지 다양하게 선택하였다. 이를 통해 Ext4 파일시스템이 도입되지 않은 시스템에서 파일 삭제 시 메타데이터 변화와 Ext4 파일시스템이 도입된 후의 파일 삭제 시 메타데이터 변화를 비교하고자 하였다. 또한 Redhat의 경우 Server로 주요 사용되는 배포판이다. 해당 배포판에서는 같은 버전의 커널을 사용하여 Redhat Desktop 버전과 Redhat Server 버전의 메타데이터 변화를 비교하고자 하였다. 각 배포판별 다운로드한 버전에 해당하는 커널 버전은 Table 5.와 같고, 분석을 진행한 환경은 Table 6.과 같다.

Table 5. Version used in each distribution

Distribution	Version	Kernel
Ubuntu	14.04	4.4.0-31
	8.04	2.6.24-16
Redhat	7.4 (Desktop)	3.10.0-693
	7.4 (Server)	3.10.0-693
Mint	18.3	4.10.0-38
	17	3.13.0-24

Table 6. Experiment environment

	Environment
OS	Windows10 Pro
RAM	16GB
HDD	SSD 256GB
CPU	I5-6600
GPU	GTX1060
VMware RAM	2GB
VMware HDD	20GB
VMware CPU	1 cpu processor

4.2 덤프 이미지 구성 및 분석 방법

다양한 커널 버전의 메타데이터 분석과 고용량 이미지와 저용량 이미지 간의 차이를 분석하기 위해 100MB와 10GB의 Ext2/3 이미지 네 개를 사용하며. 이미지는 비교를 위해 동일한 데이터 셋을 넣고 삭제 전/후 두 가지를 생성한다. 결과적으로 Table 7.과 같이 각 배포판마다 총 8개의 덤프 이미지가 생성된다. 미리 구성된 데이터셋은 모든 배포판에 동일하게 사용되며, Table 8.과 같이 구성되어있다. 데이터셋은 일상생활에서 자주 사용하는 확장자들을 사용해 구성하였다. 총 14개의 확장자를 가지며, 26개의 파일로 구성되고, 총 크기는 475MB이다. 단, 100MB의 이미지 생성에 사용되는 데이터 셋은 용량 문제로 인해 Table 8.에서 mp4 확장자를 갖는 동영상 파일 1개를 제거하여 사용했다.

분석은 생성한 다양한 커널 버전의 이미지를 이용해 SuperBlock에 대한 분석을 진행하여 Group Descriptor의 위치를 알아내며, 최종적으로 Inode Bitmap과 Inode Table의 메타데이터에 대한 분석을 하는 순서로 진행하였다. 분석을 통해 얻은 메타데이터의 비교는 Fig.6.과 같이 이미지 크기별, 파일시스템별로 나누어 4개의 과정으로 진행하였다.

Table 7. Information of dump image

File System	Dump Image
Ext2/Ext3	100MB Before Image
	100MB After Image
	10GB Before Image
	10GB After Image

Table 8. Data Set Configuration

Image Size	10GB	100MB
Number of files	26	25
Number of Extension	14	14
Type of extension	.pdf .png .py .txt .csv .mp4 .jpg .hwp .xls .pptx .zip .docx .doe .xlsx	
Size	475MB	83.8MB

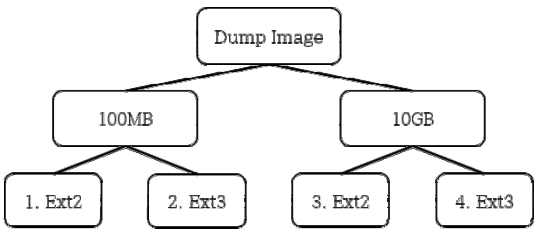

Fig. 6. Dump image comparison and analysis sequence

4.3 커널별 Superblock 비교

앞선 3.1절의 Table 1.과 같이 Superblock은 10개의 필드를 가지고 있었고 기존 내용과 대부분 일치하였다. 100MB의 모든 이미지를 대상으로 분석을 진행한 결과 Inode의 크기가 0x80, 10GB의 이미지는 Inode의 크기가 0x100으로 차이점을 보였다. 또한, Ext2/3에 따라 s_feature_compat 필드의 값에 차이가 있었으며, s_feature_incompat 필드의 경우 Ext4를 대상으로 분석하지 않았으므로 0x0040의 값은 존재하지 않았다.

4.4 커널별 Group Descryptor 비교

앞선 3.2절의 Table 2.와 같이 Group Descryptor에서 주요한 정보를 가진 3개의 필드를 가지고 있었다. 해당 필드의 값에는 커널 버전/배포판 별 차이는 존재하지 않았으며, 이미지의 크기에 따른 차이만 존재하였다. 100MB 이미지의 경우 순서대로 0x0103, 0x0104, 0x0105이며, 10GB 이미지의 경우 순서대로 0x0281, 0x0282, 0x0283의 값을 가졌다.

4.5 커널별 Inode Bitmap 비교

Inode Bitmap 분석을 진행하면서 Ubuntu 8.04의 Ext3 이미지를 제외한 모든 이미지에서 Block Group 0에 실제 파일 수만큼 모든 파일의 정보가 저장돼 있는 것을 확인하였다. Ubuntu 8.04의 Ext3 이미지에서는 Block Group 0에 실제로 저장된 파일 수에 비해 Inode Bitmap에 작성된 정보의 수가 적었다. 이미지 크기별 Block Group 0에 기록된 파일의 수는 Table 9.와 같았다. 100MB 이미지의 경우 나머지 Block Group 3과 Block Group 5에 나머지 22개의 파일에 대한 아이노드 정보가 기록돼 있었고, 10GB 이미지의 경우 26개의 파일에 대한 아이노드 정보가 Block Group 3과 Blcok Group 5에 기록돼 있었다.

덤프 이미지를 미리 구성한 데이터셋만을 넣어 만들었기 때문에 모든 커널 버전에서 동일한 결과가 나올 것이라 추정했지만 Ubuntu8.04(2.6.24)에서는 다른 결과가 나왔다. 낮은 버전의 커널에서 위와 같은 결과가 나타나는 것으로 추정돼 Ubuntu9.04(2.6.31), Ubuntu10.04(2.6.32) 환경에서 추가적인 실험을 진행해 본 결과 같은 결과를 얻었다.

Table 9. Comparison of Inode Bitmap of Block Group 0 in Ubuntu 8.04 Ext3 Image

Image Size	100MB	10GB
Number of Files	25개	26개
Number of Inodes	3개	0개

4.6 커널별 Inode Table 비교

본 논문에서 진행한 Inode Table 분석은 앞 장에서 설명한 메타데이터 구조에 기초하여 진행하였으며, 파일 복구에 필요한 새로운 유의미한 값을 얻기 위해 기존에 알려진 메타데이터 필드 외의 필드도 분석하였다. 분석을 진행하면서 모든 커널 버전에 서 100MB의 이미지는 한 inode에 128bytes를 할당하고 있음을 알아내었고, 10GB의 이미지에서는 Ubuntu 8.04의 Ext3 이미지(128bytes)를 제외하고 한 inode에 256bytes를 할당하는 것을 알아내었다.

Table 10.은 일반적으로 알려진 Ext 파일시스템에서 파일을 삭제할 때 발생하는 Inode Table의 메타데이터 변화를 나타낸 것이다.

Table 10. Metadata changes when files are deleted(5)

Field	Metadata changes
Size	Change to 0
Change Time	Change to deleted time
Modification Time	Change to deleted time
Delete Time	Change to deleted time
Block Pointer	Change to 0

4.6.1 100MB - Ext2

100MB의 Ext2 이미지의 분석/비교를 진행한 결과 파일 삭제 시 발생하는 메타데이터 변화가 Table 13.과 같이 발생했다. 3.0 이하의 커널 버전을 사용하는 배포판(Ubuntu8.04)에서는 Table 10.과 달리 메타 수정 시간과 삭제 시간만이 변화하

였고, 추가로 Link Count 필드 총 세 가지 필드만 변화하는 것을 확인하였다. 3.0 이상의 커널 버전을 사용하는 배포판은 Table 10.의 메타데이터 필드의 변화와 동일했으며, 추가로 Link Count와 Block Count 필드가 변화하는 것을 발견하였다. 3.0 이상의 커널 버전에서는 파일 삭제 시 Block Pointer 값들이 지워지는 것을 발견하였다. Ext2 파일시스템은 저널링 기법을 사용하지 않기 때문에 Block Pointer가 지워지면 파일의 복구가 불가능하다. 따라서 Ubuntu 8.04 이미지에서만 25개의 파일을 모두 복구할 수 있었고 다른 이미지에서는 파일의 삭제 여부만 알 수 있을 뿐 파일 복구는 불가능했다.

4.6.2 100MB - Ext3

100MB의 Ext3 이미지의 분석/비교를 진행한 결과 파일 삭제 시 발생하는 메타데이터 변화가 Table 13.과 같이 발생했다. Table 10.의 메타데이터 변화와 거의 동일하며 추가로 Link Count와 Block Count 필드에서 변화가 발생했다.

Ext3 파일시스템은 Ext2 파일시스템과 달리 저널링 기법을 사용하므로 Block Pointer가 지워져도 저널 영역에 있는 아이노드의 백업을 이용해 파일을 복구할 수 있다. 이 방법을 이용하기 위해 저널 영역을 분석해 본 결과 Ubuntu 14.04, Mint 18.3, Redhat Server의 경우 저널 영역에 20개의 아이노드에 대한 정보가 존재하였고 나머지 5개의 아이노드에 대한 정보는 존재하지 않았다. 5개의 아이노드는 앞에서부터 기록된 순서로 5개였다. 결과적으로 해당 이미지에서는 총 20개의 파일을 복구할 수 있다.

Ubuntu 8.04, Mint 17, Redhat Desktop은 모든 파일에 대한 아이노드의 백업이 저널 영역에 남아있어서 25개의 파일 모두 복구할 수 있었다. 단, Ubuntu 8.04의 경우 Block Group 0/3/5에서 모든 파일의 Inode 정보를 얻을 수 있었고, 나머지는 Block Group 0에서 모든 파일의 Inode 정보를 얻을 수 있었다.

4.6.3 10GB - Ext2

10GB의 Ext2 이미지의 분석/비교를 진행한 결과 파일 삭제 시 발생하는 메타데이터 변화가 Table 14.와 같이 발생했다. 3.0 버전 이전의 커널에서는 100MB Ext2 이미지의 메타데이터 변화와 동일하였으며, 3.0 버전 이후의 커널에서는 한 inode에 할당하는 크기가 256bytes이므로 100MB 이미지와 다르게 Extra Time 필드가 존재했고, 해당 필드에 차이가 발생했으며 나머지 필드는 100MB Ext2 이미지의 메타데이터 변화와 같았다. Extra change Time과 Extra Modification Time 필드에서 추가적인 메타데이터 변화가 발생하였으며 발생한 차이는 Table 11.과 같은 의미를 해석할 수 없는 값으로 나타났다.

100MB의 Ext2 이미지 분석과 마찬가지로 Ubuntu 8.04를 제외한 Ext2 파일시스템에서 Block Pointer 필드 값들이 0으로 지워지는 것을 발견하였다. Ubuntu 8.04에서는 26개의 파일을 모두 복구할 수 있었고, 이를 제외한 이미지에서는 파일의 삭제 여부만 알 수 있을 뿐 파일 복구는 불가능했다.

Table 11. Extra Time field value of Ext2 file system 10GB image

kernel	field	before deletion	after deletion
4.4.0-31	extra ctime	Sunday, November 24, 2047 8:27:08	Thursday, June 1, 2090 10:20:04
	extra mtime		
4.10.0-38	extra ctime	Sunday, November 19, 2017 3:01:28	Saturday, August 3, 2058 9:11:20
	extra mtime		
3.13.0-24	extra ctime	Tuesday, May 27, 2070 00:32:08	Friday, August 30, 1991 10:23:00
	extra mtime		
3.10.0 -693 (Server)	extra ctime	Thursday, September 4, 2008 16:19:08	Friday, September 6, 2075 3:09:00
	extra mtime		
3.10.0 -693 (Desktop)	extra ctime	Thursday, January 14, 2049 8:28:48	Monday, December 13, 2038 9:06:52
	extra mtime		

4.6.4 10GB - Ext3

10GB의 Ext3 이미지의 분석/비교를 진행한 결과 파일 삭제 시 발생하는 메타데이터 변화가 Table 14.와 같이 발생했다. 10GB의 Ext2 이미지와 동일한 차이가 발생하였고, Ubuntu8.04를 제외한 모든 커널 버전에서 Extra Time 필드의 차이가 발생하였으며 Table 12.와 같은 의미를 해석할 수 없는 값으로 나타났다.

분석 결과 Block Pointer가 지워져도 Ext3 파일시스템은 저널링 기법을 사용하므로 저널 영역에 작성돼있는 아이노드의 백업을 이용해 파일을 복구할 수 있었고, 모든 버전의 커널에서 26개의 파일 모두 복구에 성공하였다. 단, Ubuntu 8.04의 경우 Block Group 0/3/5에서 모든 파일의 Inode 정보를 얻을 수 있었고, 나머지는 Block Group 0에서 모든 파일의 Inode 정보를 얻을 수 있었다.

Table 12. Extra Time field value of Ext3 file system 10GB image

kernel	field	before deletion	after deletion
4.4.0-31	extra ctime	Friday, November 4, 1972 6:15:16	Thursday, October 13, 2044 18:18:40
	extra mtime	Friday, June 22, 2018 22:41:12	
4.10.0-38	extra ctime	Sunday, September 1, 2041 8:26:16	Wednesday, November 26, 2008 00:26:32
	extra mtime		
3.13.0-24	extra ctime	Friday, September 13, 2041 11:16:20	Friday, August 8, 2059 5:53:52
	extra mtime		
3.10.0 -693 (Server)	extra ctime	Wednesday, December 30, 2015 9:03:04	Monday, June 13, 2005 3:45:20
	extra mtime		
3.10.0 -693 (Desktop)	extra ctime	Monday, January 7, 2008 13:26:12	Saturday, October 4, 2087 9:00:36
	extra mtime		

4.7 실험 결과 분석

본 장에서 커널 버전별/배포판별 차이에 따른 메타데이터 분석을 진행했다. Ubuntu 8.04(3.0 이전)의 이미지를 제외하면 파일시스템 종류(Ext2/3)와 이미지 크기(100MB/10GB)에 상관없이 파일 크기, 메타 수정 시간, 파일 수정 시간, 삭제 시간, Link Count, Block Count, Block Pointer 총 7개의 필드가 변경되는 것을 확인했다. Ubuntu 8.04 배포판의 경우 Ext2 파일시스템을 사용할 때 다른 이미지와 많은 차이를 보였다. 다른 커널 버전을 사용하는 배포판들과 달리 메타 수정 시간, 삭제 시간, Link Count 필드만 변경되었다. 이로 인해 다른 배포판으로 생성한 Ext2 이미지에서는 Block Pointer 필드가 지워져 파일의 삭제 여부만 확인 가능했던 것에 반해 해당 이미지에서는 파일을 복구해낼 수 있었다.

기존 포렌식 수사에서 주로 사용하는 메타데이터 필드는 파일의 크기, 삭제 시간, 메타 수정 시간, 파일 수정 시간, Block Pointer인데 실험을 진행하면서 메타데이터를 분석한 결과 파일 삭제 시 Link Count와 Block Count 필드가 포렌식 수사에 사용할 수 있는 의미 있는 변화를 갖는 것을 확인하였다[5]. Link Count는 해당 파일이 몇 개의 Link를 가지고 있는지를 나타내는 값이다. Hard Link와 Symbolic Link를 포함하며, 파일이 존재하면 최소 1의 값을 갖는다. 그리고 Link Count 필드가 파일 삭제 시 1 이상의 값에서 0으로 변경되는 것을 확인했다. Block Count는 해당 파일이 차지하는 전체 블록의 수를 의미하는 값이다. Block Count 필드 또한 파일이 삭제되지 않았을 때 1 이상의 값을 가지며 파일이 삭제되면 0으로 변경되는 것을 확인하였다.

분석한 이미지를 통해 파일을 복원한 결과 Ext2의 경우 Ubuntu 8.04 이미지에서만 삭제된 파일 복구에 성공했다. 파일 삭제 시 Ubuntu 8.04에서만 Block Pointer가 지워지지 않았고 나머지 배포판에서는 Block Pointer가 지워졌다. Ext2 파일시스템의 경우 저널링 기법을 사용하지 않기 때문에 Block Pointer가 지워지면 파일을 복구할 수 없게 된다. 나머지 이미지에서는 파일이 삭제됐다는 것만 알아낼 수 있을 뿐 파일 복구는 불가능하였다. 이는 커널 버전이 높아짐에 따라 저널링 기법을 사용하고 점유율이 높은 파일시스템인 Ext 3/4에 주 초점이

Table 13. Metadata comparison of inode table before and after deletion in 100MB image

File system	Ext2		Ext3	
Deletion	before	after	before	after
Ubuntu 14.04 (kernel ver : 4.4.0-31)				
Size	14 10 00 00	**00 00 00 00**	14 10 00 00	00 00 00 00
Change Time	36 5B 2E 5A	5B 28 2F 5A	3F 5B 2E 5A	68 28 2F 5A
Modification Time	36 5B 2E 5A	**5B 28 2F 5A**	3F 5B 2E 5A	68 28 2F 5A
Deletion Time	00 00 00 00	5B 28 2F 5A	00 00 00 00	68 28 2F 5A
Link Coun	01 00	00 00	01 00	00 00
Block Count	0A 00	**00 00**	0A 00	00 00
Block Pointer	00 01 04 00 …	**00 00 00 00 …**	00 01 04 00 …	00 00 00 00 …
Mint 18.3 (kernel ver : 4.10.0-38)				
Size	14 10 00 00	**00 00 00 00**	14 10 00 00	00 00 00 00
Change Time	8A A7 30 5A	6A F2 39 5A	84 A7 30 5A	6A F2 39 5A
Modification Time	8A A7 30 5A	**6A F2 39 5A**	84 A7 30 5A	6A F2 39 5A
Deletion Time	00 00 00 00	6A F2 39 5A	00 00 00 00	6A F2 39 5A
Link Count	01 00	00 00	01 00	00 00
Block Count	0A 00	**00 00**	0A 00	00 00
Block Pointer	01 04 00 00 …	**00 00 00 00 …**	01 04 00 00 …	00 00 00 00 …
Mint 17 (kernel ver : 3.13.0-24)				
Size	14 10 00 00	**00 00 00 00**	14 10 00 00	00 00 00 00
Change Time	98 91 2F 5A	C1 97 2F 5A	AB 90 2F 5A	CE 97 2F 5A
Modification Time	98 91 2F 5A	**C1 97 2F 5A**	AB 90 2F 5A	CE 97 2F 5A
Deletion Time	00 00 00 00	C1 97 2F 5A	00 00 00 00	CE 97 2F 5A
Link Count	01 00	00 00	01 00	00 00
Block Count	0A 00	**00 00**	0A 00	00 00
Block Pointer	01 04 00 00 …	**00 00 00 00 …**	01 04 00 00 …	00 00 00 00 …
Redhat Server (kernel ver : 3.10.0-693)				
Size	14 10 00 00	**00 00 00 00**	14 10 00 00	00 00 00 00
Change Time	4F A0 30 5A	EE A0 30 5A	D6 55 30 5A	36 78 30 5A
Modification Time	4F A0 30 5A	**EE A0 30 5A**	D6 55 30 5A	36 78 30 5A
Deletion Time	00 00 00 00	EE A0 30 5A	00 00 00 00	36 78 30 5A
Link Count	01 00	00 00	01 00	00 00
Block Count	0C 00	**00 00**	0C 00	00 00
Block Pointer	01 04 00 00 …	**00 00 00 00 …**	01 04 00 00 …	00 00 00 00 …
Redhat Desktop (kernel ver : 3.10.0-693)				
Size	14 10 00 00	**00 00 00 00**	14 10 00 00	00 00 00 00
Change Time	FD 3B 30 5A	9E 41 30 5A	F1 3B 30 5A	A1 41 30 5A
Modification Time	FD 3B 30 5A	**9E 41 30 5A**	F1 3B 30 5A	A1 41 30 5A
Deletion Time	00 00 00 00	9E 41 30 5A	00 00 00 00	A1 41 30 5A
Link Count	01 00	00 00	01 00	00 00
Block Count	0C 00	**00 00**	0C 00	00 00
Block Pointer	01 04 00 00 …	**00 00 00 00 …**	01 04 00 00 …	00 00 00 00 …
Ubuntu 8.04 (kernel ver : 2.6.24-16)				
Size	14 10 00 00		14 10 00 00	00 00 00 00
Change Time	5D 48 2F 5A	04 89 2F 5A	9A 04 3A 5A	B0 23 3A 5A
Modification Time	5D 48 2F 5A		9A 04 3A 5A	B0 23 3A 5A
Deletion Time	00 00 00 00	04 89 2F 5A	00 00 00 00	B0 23 3A 5A
Link Count	01 00	00 00	01 00	00 00
Block Count	0A 00		0A 00	00 00
Block Pointer	01 12 00 00 …		03 68 00 00 …	00 00 00 00 …

Table 14. Metadata comparison of inode table before and after deletion in 10GB image

File system	Ext2		Ext3	
Deletion	before	after	before	after
Ubuntu 14.04 (kernel ver : 4.4.0-31)				
Size	14 10 00 00	**00 00 00 00**	14 10 00 00	00 00 00 00
Change Time	5F 57 2E 5A	62 28 2F 5A	84 5B 2E 5A	65 28 2F 5A
Modification Time	5F 57 2E 5A	**62 28 2F 5A**	84 5B 2E 5A	65 28 2F 5A
Deletion Time	00 00 00 00	62 28 2F 5A	00 00 00 00	65 28 2F 5A
Link Count	01 00	00 00	01 00	00 00
Block Count	10 00	**00 00**	10 00	00 00
Block Pointer	00 06 00 00 …	**00 00 00 00 …**	00 06 04 00 …	00 00 00 00 …
Mint 18.3 (kernel ver : 4.10.0-38)				
Size	14 10 00 00	**00 00 00 00**	14 10 00 00	00 00 00 00
Change Time	A8 A7 30 5A	6A F2 39 5A	94 A7 30 5A	6B F2 39 5A
Modification Time	A8 A7 30 5A	**6A F2 39 5A**	94 A7 30 5A	6B F2 39 5A
Deletion Time	00 00 00 00	6A F2 39 5A	00 00 00 00	6B F2 39 5A
Link Count	01 00	00 00	01 00	00 00
Block Count	10 00	**00 00**	10 00	00 00
Block Pointer	00 06 04 00 …	**00 00 00 00 …**	00 06 04 00 …	00 00 00 00 …
Mint 17 (kernel ver : 3.13.0-24)				
Size	14 10 00 00	**00 00 00 00**	14 10 00 00	00 00 00 00
Change Time	A7 91 2F 5A	C7 97 2F 5A	38 92 2F 5A	CB 97 2F 5A
Modification Time	A7 91 2F 5A	**C7 97 2F 5A**	38 92 2F 5A	CB 97 2F 5A
Deletion Time	00 00 00 00	C7 97 2F 5A	00 00 00 00	CB 97 2F 5A
Link Count	01 00	00 00	01 00	00 00
Block Count	10 00	**00 00**	10 00	00 00
Block Pointer	00 06 04 00 …	**00 00 00 00 …**	00 06 04 00 …	00 00 00 00 …
Redhat Server (kernel ver : 3.10.0-693)				
Size	14 10 00 00	**00 00 00 00**	14 10 00 00	00 00 00 00
Change Time	33 56 30 5A	3A 78 30 5A	E1 55 30 5A	36 78 30 5A
Modification Time	33 56 30 5A	**3A 78 30 5A**	E1 55 30 5A	36 78 30 5A
Deletion Time	00 00 00 00	3A 78 30 5A	00 00 00 00	36 78 30 5A
Link Count	01 00	00 00	01 00	00 00
Block Count	10 00	**00 00**	10 00	00 00
Block Pointer	00 06 04 00 …	**00 00 00 00 …**	00 06 04 00 …	00 00 00 00 …
Redhat Desktop (kernel ver : 3.10.0-693)				
Size	14 10 00 00	**00 00 00 00**	14 10 00 00	00 00 00 00
Change Time	06 3C 30 5A	9E 41 30 5A	24 3C 30 5A	A1 41 30 5A
Modification Time	06 3C 30 5A	**9E 41 30 5A**	24 3C 30 5A	A1 41 30 5A
Deletion Time	00 00 00 00	9E 41 30 5A	00 00 00 00	A1 41 30 5A
Link Count	01 00	00 00	01 00	00 00
Block Count	10 00	**00 00**	10 00	00 00
Block Pointer	00 06 04 00 …	**00 00 00 00 …**	00 06 04 00 …	00 00 00 00 …
Ubuntu 8.04 (kernel ver : 2.6.24-16)				
Size	14 10 00 00		14 10 00 00	00 00 00 00
Change Time	28 49 2F 5A	0C 89 2F 5A	A4 04 3A 5A	B1 23 3A 5A
Modification Time	28 49 2F 5A		A4 04 3A 5A	B1 23 3A 5A
Deletion Time	00 00 00 00	0C 89 2F 5A	00 00 00 00	B1 23 3A 5A
Link Count	01 00	00 00	01 00	00 00
Block Count	10 00		10 00	00 00
Block Pointer	00 58 00 00 …		98 01 00 01 …	00 00 00 00 …

맞춰지기 때문에 Ext2 파일시스템에서 파일 삭제 시 발생하는 메타데이터 변화가 Ext3/4와 유사한 것으로 추정된다.

Ext3의 경우 10GB 이미지에서는 모든 배포판에서 파일의 복구가 가능하나, 2버전 대의 커널에서 일반적인 파일시스템과 Inode Bitmap과 Inode Table에서 다른 양상을 보였다. 같은 데이터를 넣고 진행한 실험에서 3/4버전 대의 커널 버전에서는 Block Group 0에 모든 Inode 정보가 기록돼있는 반면, 2.6.24/2.6.31/2.6.32 커널 버전의 경우 Block Group 0뿐만 아니라 Block Group 3/5에 Inode 정보가 나눠 저장돼 있었다. 실제 포렌식 수사에서 이러한 차이를 알지 못하고 분석을 진행한다면 파일의 삭제 여부나 존재 여부를 알아낼 수 없으므로 파일의 복구 가능성이 줄어들 것으로 추정된다.

100MB의 이미지에서는 모든 이미지에서 파일을 복구할 수 있었지만 복구에 성공한 파일의 개수에서 차이를 보였다. Ubuntu 14.04, Mint 18.3, Redhat Server 그룹은 25개 파일 중 20개 파일의 복구에 성공했다, Mint 17, Redhat Desktop 그룹은 모든 파일의 복구에 성공했다. 항상 이러한 결과가 나타나는 것인지 확인을 위해 다양한 데이터셋을 이용한 여러 번의 실험을 진행한 결과 Ubuntu 14.04, Mint 18.3, Redhat Server 그룹에서는 저널 영역에 Fig.7.과 같이 항상 12-16번 아이노드의 정보가 존재하지 않는 것을 확인하였다. 이는 파일시스템의 크기가 100MB로 작아서 나타나는 현상으로 추정된다. 일반적으로 저널은 파일시스템 크기에 따라 변화하고, 저널의 크기가 협소한 경우 파일 복구가 어려울 수 있다. 하지만 위 실험을 통해 커널 버전에 따라서도 파일시스템이 그 크기에 비해 많이 활용되는 경우 복구에 지대한 영향을 줄 수 있는 차이를 보이는 것을 확인하였다.

실제 포렌식 수사에서 파일 삭제 여부를 분석할 때 실험을 통해 발견한 Link Count와 Blcok Count 필드를 추가적으로 사용한다면 파일의 삭제 여부를 판단하는데 도움이 될 것이라 생각된다. 또한, 실험을 진행하여 발견한 리눅스 배포판/커널 버전별 메타데이터 변화의 차이를 알지 못한다면 본 논문에서 사용한 3/4버전 대의 커널을 사용하는 Ext2 파일시스템에서 삭제된 파일의 복구가 불가능하다는 것을 알 수 있었다.

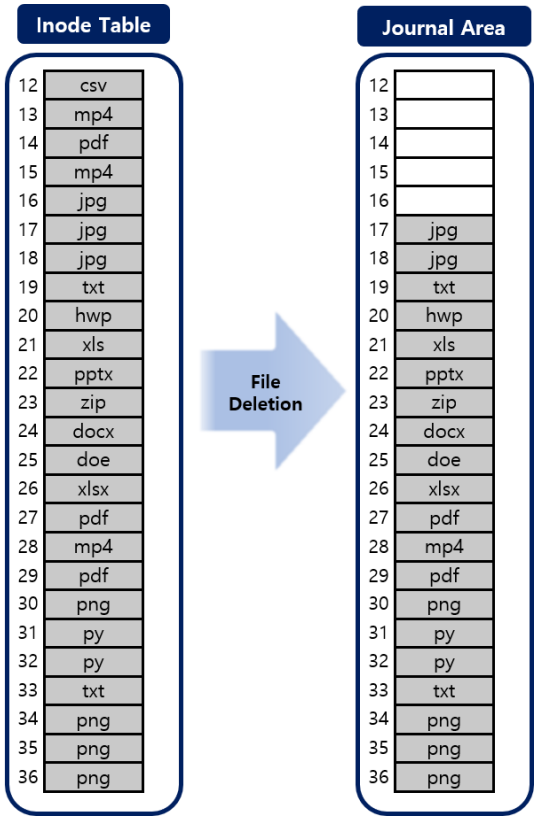

Fig. 7. Ext3 100MB image's inode information in Ubuntu14.04, Mint18.3, Rehat Server

V. 결 론

본 논문에서는 커널 버전에 따른 파일 삭제 시 발생하는 메타데이터의 변화에 대해 다뤘다. 실험을 통해 커널 버전 별 메타데이터의 변화를 비교·분석하였고, 모든 커널 버전에서 공통적으로 변화하는 메타데이터 필드, 커널 버전 별 차이가 있는 메타데이터 필드, 이미지 크기별 변화에 차이가 있는 메타데이터 필드를 알아냈다. 새롭게 알아낸 메타데이터 필드를 적극적으로 활용해 파일의 삭제 여부를 판단하였고 삭제된 파일들을 복구하기 위해 Ext3 파일시스템에서는 저널 영역을 분석하였다. 대부분의 배포판에서 저널 영역에 삭제된 모든 파일에 대한 아이노드의 백업이 존재하였고 몇몇 배포판에서는 Inode Table 앞에서부터 기록된 5개의 아이노드에 대한 백업이 존재하지 않았다. 다양한 데이터셋을 이용해 실험을 해본 결과 앞에서 기록된 5개의 아이노드에 대한 백업은 항상 저널 영역에 존재하지 않는 것을 확인했다.

파일 삭제 시 발생하는 메타데이터 변화량은 같은 용량의 이미지의 경우 Ext2 파일시스템과 Ext3 파일시스템이 유사하였다. 100MB 이미지의 경우 메타데이터 변화량이 Ext2 파일시스템은 평균 6.6개 필드, Ext3 파일시스템은 평균 7.3개 필드였고, 10GB 이미지의 경우 Ext2 파일시스템은 평균 8.5개 필드, Ext3 파일시스템은 평균 8.6개 필드이었다.

파일 삭제 및 복구에 관련된 주요 파일시스템 메타데이터 변화를 분석한 결과 리눅스 커널 3.0 이상과 미만인 경우에 차이가 발생함을 보였다. Ubuntu의 경우 8.04 버전은 3.0 이하 버전의 커널을 사용하며 이후 배포판은 3.0 이상 버전의 커널을 사용한다. Ubuntu 8.04 배포판은 Ext2 파일시스템에서 다른 배포판과 달리 파일 삭제 시 삭제된 파일의 아이노드 내 Block Pointer가 지워지지 않는다. 다른 배포판에서는 일반적으로 알려진 바와 같이 Block Pointer를 포함한 일체의 메타데이터가 삭제되었다. 하지만 Ext2는 저널링 기법을 사용하지 않기 때문에 Block Pointer가 지워진 리눅스 커널 3.0 이후의 파일시스템에서는 메타데이터를 통한 파일 복구가 힘들다. 이는 커널 버전이 높아짐에 Ext3/4 파일시스템에 주 초점이 맞춰져 Ext2 파일시스템에서 파일 삭제 시 발생하는 메타데이터 변화가 Ext3/4와 유사한 것으로 추정된다. 또한 Ubuntu 8.04의 Ext3 이미지의 경우 아이노드에 대한 정보를 나타내는 Inode Bitmap과 Inode Table의 작성에 3개의 Block Group을 사용하였다. 일반적인 작성 방법과 차이가 있는 것을 확인했다. 해당 커널 버전에서만 발생하는 현상인지에 대한 확인을 위해 추가로 Ubuntu 9.04와 Ubuntu 10.04에서 실험을 진행한 결과 같은 결과를 얻었다.

실험을 통해 얻은 메타데이터의 변화를 토대로 전 세계 수사기관은 Ext2/3 파일시스템을 활용하는 다양한 시스템에서 파일의 삭제 여부를 더 정확하게 판단할 수 있을 것이며, 기존에 삭제 여부를 판단하기 어려웠던 수많은 파일들에 대한 커버리지를 확장시킬 수 있을 것이다.

향후 연구로는 Ext4와 다양한 배포판이 분석 대상으로 활용될 수 있다.

References

[1] Soowoong Eo, Wooyeon Jo, Seokjun Lee, and Taeshik Shon, "A phase of deleted file recovery for digital forensics research in Tizen," 5th IT Convergence and Security (ICITCS), pp. 1-3, Aug. 2015.

[2] Dimitrios Kasiaras, Thomas Zafeiropoulos, Nathan Clarke, and Georgios Kambourakis, "Android forensics: Correlation analysis," 9th Internet Technology and Secured Transactions (ICITST), pp. 157-162. Dec. 2014.

[3] Qian Li, Xueli Hu, and Hao Wu, "Database management strategy and recovery methods of Android," 5th Software Engineering and Service Science (ICSESS), pp. 727-730, Jun. 2014.

[4] Dohyun Kim, Jungheum Park, and Sangjin Lee, "File Carving for Ext4 File System on Android OS," Journal of the Korea Institute of Information Security & Cryptology 23(3), pp. 417-429, Jun. 2013.

[5] Seokjun Lee and Taeshik Shon, "Improved deleted file recovery technique for Ext2/3 filesystem," The Journal of Supercomputing, vol. 70, no. 1, pp. 20-30, Oct. 2014.

[6] Zhi Li, Bin Xi, and Shunxiang Wu, "Digital forensics and analysis for Android devices," 11th Computer Science & Education (ICCSE), pp. 496-500, Aug. 2016.

[7] KD Fairbanks, "An analysis of Ext4 for digital forensics," Digital investigation, vol. 9, pp. 118-130, Aug. 2012.

[8] Fahad Alanazi and Andrew Jones, "The Value of Metadata in Digital Forensics," European Intelligence and Security Informatics Conference (EISIC), pp. 182-182, Sep. 2015.

[9] Andreas Dewald and Sabine Seufert, "AFEIC: Advanced forensic Ext4 inode

carving," Digital Investigation, vol. 20, pp. 83-91, Mar. 2017.
〔10〕 Akshara Ravi, T. Raj Kumar, and Angelo Renju Mathew, "A method for carving fragmented document and image files," Advances in Human Machine Interaction (HMI), pp. 1-6, Mar. 2016.
〔11〕 Mohammed Alhussein, Avinash Srinivasan, and Duminda Wijesekera, "Forensics filesystem with cluster-level identifiers for efficient data recovery," Internet Technology And Secured Transactions, pp. 411-415, Dec. 2012.
〔12〕 Gianpaolo Macario, Marco Torchiano, and Massimo Violante, "An in-vehicle infotainment software architecture based on google android," 2009 IEEE International Symposium on Industrial Embedded Systems, pp. 257-260, Aug. 2009.

〈저 자 소 개〉

신 영 훈(Yeonghun Shin) 학생회원
2019년: 아주대학교 사이버보안학과 졸업(학사)
2019년~현재: 아주대학교 컴퓨터공학과 재학(통합)
〈관심분야〉 디지털 포렌식, IoT 보안, 네트워크 포렌식

조 우 연(Woo-yeon Jo) 학생회원
2015년: 아주대학교 정보컴퓨터공학과 졸업(학사)
2015년~현재: 아주대학교 컴퓨터공학과 재학(통합)
〈관심분야〉 디지털 포렌식, 네트워크 시각화, 제어시스템 보안, IoT 보안, 개인정보보호

손 태 식 (Taeshik Shon) 종신회원
2000년: 아주대학교 정보및컴퓨터공학부 졸업(학사)
2002년: 아주대학교 정보통신전문대학원 졸업(석사)
2005년: 고려대학교 정보보호대학원 졸업(박사)
2004년~2005년: University of Minnesota 방문연구원
2005년~2011년: 삼성전자 통신· DMC 연구소 책임연구원
2017년~2018년: Illinois Institute of Technology 방문교수
2011년~현재: 아주대학교 정보통신대학 사이버보안학과 교수
〈관심분야〉 ICS/SCADA, DFIR, Anomaly Detection