ISSN(Print) : 1976-5304, ISSN(Online) : 2713-931X
https://kdfs.jams.or.kr, DOI : 10.22798/kdfs.2022.16.2.1

NTFS $LogFile를 이용한 MS Office 문서 열람 및 작성 흔적 조사에 관한 연구

정 재 훈*, 박 상 준**
대검찰청 정보통신과(검찰주사보)*, 서울대학교 수리정보과학과(객원교수)**

A Study on Investigating Reading and Writing Traces of MS Office Documents Using NTFS $LogFile

Jaehoon Jeong*, Sangjoon Park**
Supreme Prosecutor's Office (Seven grade Prosecutor)*,
Dept. of Mathematical Information Science, Seoul National University (Visiting Professor)**

요 약

윈도우 시스템에 대한 포렌식 수사에서 문서 파일을 읽거나 수정하거나 접근한 흔적을 조사할 때 가장 일반적으로 많이 이용되는 것은 바로가기 파일(.LNK)로 구성된 최근사용문서와 점프리스트이다. 그러나, 바로가기 파일이나 점프리스트는 잘 알려져 있어 삭제되기 쉬우며, 기록이 남아 있다 하더라도 가장 최근 사용 기록만 존재하고 어느 정도 수정 작업이 있었는지 확인할 수 없는 문제점이 있다. 한편 $LogFile은 저장된 레코드를 보유하는 기간 동안 해당 볼륨에서 MS Office 문서 파일의 오픈, 수정, 저장, 종료와 관련된 모든 기록을 축적하여 가지고 있으며, 수정 및 저장시 파일 크기 변경 기록도 축적하여 가지고 있기 때문에 해당 파일이 어느 시점에 오픈되었으며 어느 정도 변경되었는지 확인할 수 있는 장점이 있다. 그러나, 운영체제가 설치된 볼륨 C:\에 있는 $LogFile은 윈도우 시스템에 의한 변경 사항이 너무 많아 3~4시간 정도의 기록밖에는 유지하지 못하여 포렌식 수사에 사용하기에는 한계가 있다. 반면, $LogFile은 모든 NTFS 볼륨에 기본적으로 존재하고 볼륨 C:\가 아닌 NTFS 볼륨의 $LogFile은 3일에서 15일 정도의 기록을 유지한다. 따라서 볼륨 C:\가 아닌 NTFS 볼륨 $LogFile에서 MS Office 문서 파일의 열람 및 작성 흔적을 조사하는 것은 의미가 있다. 본 논문에서는 $LogFile을 이용하여 MS Office 문서 열람 및 작성 작업을 재구성할 수 있음을 보이고 $LogFile 분석 도구인 NTFS Log Tracker 1.7을 이용하여 MS Office 문서 열람 및 작성 흔적을 조사할 수 있음을 보이고자 한다. 또한, $LogFile 로그 레코드를 분석을 통하여 파일 수정 및 저장시 변경되는 파일의 크기를 조사할 수 있는 방법을 제시하고자 한다.

주제어 : 컴퓨터 포렌식, NTFS, $LogFile, 마이크로소프트 오피스

ABSTRACT

The most commonly used things in forensic investigations of Windows systems to investigate traces of reading, modifying, or accessing document files are the jumplists and shortcut files(*.LNK) under the folder %AppData%\Microsoft\Windows\Recent. However, the shortcut files and jumplists are well known and easy to delete, and even if records remain, there is a problem that only the most recent usage records exist and it is impossible to determine to what extent there have been modifications. Meanwhile, $LogFile has accumulated all records related to opening, modifying, saving, and closing MS Office document files during the period of holding records, and also has stored changing records of file sizes at the time of modifying and saving. However, the $LogFile in Volume C:\, where the operating system is installed, has too many changing records by the windows operating system and can maintain only a record of 3-4 hours, which is limited to use in forensic investigations. On the other hand, $LogFile exists by default on all NTFS volumes, and $LogFile of a NTFS volume other than the volume C:\ can maintains a record of 3 to 15 days. So it is meaningful to investigate the reading and writing traces of MS Office document files using $LogFile on NTFS volumes other than the volume C:\. In this paper, we show that users the process of reading and writing MS Office documents can be reconstructed using $LogFile and the traces of MS Office documents reading and writing using the $LogFile analysis tool, NTFS Log Tracker 1.7, can be investigated. In addition, it is intended to present a method to investigate changing the size of a file when modifying and saving a file using $LogFile log records.

Key Words : Computer Forensics, NTFS, $LogFile, MS Office

▪ Received 17 February 2022, Revised 18 February 2022, Accepted 27 June 2022
▪ 제1저자(First Author) : Jaehoon Jeong (Email : yldn12016@gmail.com)
▪ 교신저자(Corresponding Author) : Sangjoon Park (Email : psjoon98@gmail.com)

Ⅰ. 서 론

NTFS는 윈도우 파일 시스템으로 저널링 기능을 제공하는 $LogFile과 $UsnJrnl 2가지 로그 파일이 있다[1][2]. $LogFile은 시스템 오류나 갑작스런 전원 차단시 파일 복구 기능을 제공하기 위해 트랜잭션 로그와 같은 시스템 변경 사항에 대한 상세한 로그가 저장되며, $UsnJrnl은 파일 복구 기능은 없고 파일시스템 변경 사항만을 기록한다. $LogFile은 모든 NTFS 볼륨의 루트 디렉토리에 존재하는 반면 $UsnJrnl은 추가 설정할 수 있긴 하지만 기본적으로 볼륨 C:\Extend 디랙토리에만 존재한다. 일반적으로 $LogFile은 64MB(67,108,864바이트) 물리적 크기를 갖고 $UsnJrnl은 대략 32MB 정도의 물리적 크기를 갖고 있어 $LogFile의 물리적인 크기가 $UsnJrnl의 물리적 크기 보다 2배 정도 크지만 $LogFile은 파일 복구를 위해 보다 상세한 시스템 변경 기록을 저장하기 때문에 볼륨 C:\의 $LogFile의 경우 3~4시간 정도의 로그 기록밖에 저장하지 못하지만 $UsnJrnl은 1~3일 정도의 시스템 변경 기록을 저장한다. 볼륨 C:\$LogFile의 로그 기록 저장 기간이 짧은 이유는 윈도우 운영체제의 시스템 활동에 따른 시스탬 변경 기록이 너무 많기 때문이다. 볼륨 C:\가 아닌 다른 NTFS 볼륨의 $LogFile은 윈도우 시스템에 의한 로그 기록이 많지 않기 때문에, 사용자와 시스템 활동에 따라 차이가 있겠지만, 비교적 긴 시간 동안(3일에서 15일 정도) 로그 기록을 유지한다. 따라서 볼륨 C:\ 이외의 NTFS 볼륨에 존재하는 $LogFile을 분석하는 것은 의미가 있다.

문서 파일에 대한 접근 흔적은 파일의 메타데이터($MFT), 그리고 %UserName%\AppData\Roaming\Microsoft\Windows\Recent 아래 바로가기 파일(*.LNK)로 구성된 최근사용문서나 점프리스트(Jump Lists)를 조사하는 것이 가장 일반적인 방법이지만[3][4][5][6], 이러한 기록들은 가장 최근의 수정이나 접근 기록 하나만을 제공할 뿐 아니라, 바로가기 파일과 점프리스트 기록은 쉽게 삭제할 수 있다는 문제점이 있다. 반면, $LogFile은 시스템 파일로서 일반 윈도우 탐색기에서는 확인할 수 없고 포렌식 도구와 같은 도구를 사용하지 않는 한 사용자가 그 존재를 인식할 수 없을 뿐 아니라 삭제하기 어렵고, $LogFile이 로그 기록을 유지하는 동안에 있었던 해당 파일에 대한 모든 작업 기록(파일 열기, 수정, 저장, 종료)을 축적하여 가지고 있다. 또한, 파일의 크기와 시간 변경 기록들도 축적하여 보유하고 있기 때문에 사용자가 해당 파일을 언제 어느 정도 수준으로 변경(작성)하였는지 확인할 수 있다. 초안 문서의 존재는 파일 작성자를 특정할 때 중요한 요소이지만[7] 파일시스템의 비할당 영역에서 파일 복구가 어려운 SSD와 같은 저장매체에서 초안 문서를 찾아내는 것은 어려운 문제이다. 본 논문에서 제안하는 방법을 사용할 경우 초안 문서를 찾아낼 수는 없다 하여도 그 존재는 증명할 수 있을 것이다.

$LogFile을 컴퓨터 포렌식 수사에 이용한 기존의 연구들은 파일/디렉토리 생성, 삭제, 복사, 이동, 이름변경할 때 윈도우 시스템이 $LogFile에 기록하는 로그 레코드들에 대한 분석을 하고 그러한 행위로 시스템이 변경되기 이전 상태의 파일이나 디렉토리 흔적을 추적하는 연구들과[8][9][10]. 안티포렌식 도구를 사용하여 시간을 변조하거나 데이터를 외이핑할 때 $LogFile에 기록되는 로그를 분석하여 안티포렌식 행위 흔적을 추적하는 연구들로 분류할 수 있다[11][12][13]. 그러나, 파일 열람 및 작성(수정)에 대한 흔적은 해당 파일을 오픈하거나 수정할 때 이용되는 어플리케이션이 어떤 로그 레코드를 $LogFile에 남기느냐에 따라 다르기 때문에 해당 어플리케이션이 파일을 열람하거나 작성(수정)할 때 $LogFile에 어떤 로그 레코드를 남기는지 조사하여야 하지만, 이제까지 MS Office가 문서 파일을 열람하거나 작성할 때 $LogFile에 어떤 로그 레코드를 남기는지를 조사한 연구는 없었다.

예를 들어, jpg 파일, pdf 파일은 일반적인 jpg 뷰어와 pdf 뷰어를 이용하여 단순 열람(문서 보기)할 경우 $LogFile에서 해당 파일 열람으로 추정할만한 어떠한 로그 레코드도 찾을 수 없었다. 한컴오피스의 hwp 파일의 경우도 한컴오피스 SW로 단순 열람만 할 경우에는 해당 파일 열람으로 추정할만한 로그 레코드를 찾을 수 없다. 그러나, MS Office의 경우에는 관련 문서 파일에 대한 단순 열람시에도 파일 오픈시 고유한 형태의 임시파일이 생성되었다가 파일 열람 종료시 삭제되고 있음을 보여주는 로그 레코드들을 확인할 수 있으며, 파일에 대한 작성(수정)시에도 고유한 형태의 임시파일들이 생성되었다가 삭제되고 있음을 보여주는 로그 레코드들을 확인할 수 있다.

본 논문에서는 볼륨 C:\가 아닌 NTFS 볼륨에서 MS Office 문서 파일들(.docx, .pptx, .xlsx)을 단순 열람하거나 작성(수정)할 경우에 MS Office 프로그램이 $LogFile에 남기는 로그 레코드들을 조사하고 그러한 조사 결과를 바탕으로 사용자가 언제 해당 파일을 열람하고 수정하였는지 추적할 수 있음을 $LogFile 분석 도구인 NTFS Log Tracker 1.7을 이용하여 보이고자 한다[14]. 또한, $LogFile의 로그 레코드를 분석하여 MS Office 파일 수정 및 저장시 파일의 실질적인 크기 변화를 추적할 수 있는 방법을 제시하고자 한다.

본 논문은 모두 8개 장으로 구성된다. Ⅱ장에서는 본 논문의 연구와 관련된 선행 연구들을 조사하였으며, Ⅲ

장에서는 NTFS $LogFile을 간략하게 소개하였다. Ⅳ장에서는 $LogFile에 기록된 로그 레코드들을 이용하여 MS Office 문서 파일 열람 및 작성 과정을 재구성하는 방법을 기술하고 문서 열람 및 작성 과정에서 어떤 형태의 흔적을 남기는지 요약 정리하였다. Ⅴ장에서는 $LogFile 분석 도구인 'NTFS Log Tracker 1.7'을 사용하여 MS Office 문서 파일 열람 및 작성 흔적을 확인하였다. Ⅵ장에서는 MS Office 문서 파일 수정 과정에서 변경되는 파일의 크기를 확인하기 위하여 어느 위치에서 어떠한 로그 레코드를 추가적으로 분석하여야 하는지 제시하였다. Ⅶ장에서는 USB 저장매체내 NTFS 볼륨에서 MS Word 문서 열람 및 작성이 남기는 로그 기록이 SSD, HDD와 차이가 나는 부분에 대하여 기술하였다. Ⅷ장은 본 논문의 결론부이다.

Ⅱ. 관련 선행 연구

2009년과 2010년 김태환과 조규상은 $LogFile의 로그 레코드를 분석하여 파일 및 디렉토리 생성, 삭제, 복사, 이동, 이름변경 과정을 재구성함으로써 포렌식 수사에 이용할 수 있는 가능성을 제시하였다[8][9][10]. 2013년 조규상은 $LogFile을 이용하여 윈도우 문서 파일의 타임스탬프(파일 생성/수정/접근 시간) 위.변조를 탐지할 수 있는 방법을 제안하였다[11]. 2020년 오동빈, 박경호, 김휘강은 머신러닝 기법으로 $LogFile과 $UsnJrnl 로그를 사용하여 데이터 와이핑 오남용을 탐지하는 효율적인 방법을 제안하였다[12]. 2021년 오정훈과 황현욱은 $LogFile과 $UsnJrnl을 이용하여 시스템에서 발생하는 사용자 의심행위들을 분류하고 각 행위를 탐지할 수 있는 현실적인 방법을 제안하였다[13].

2017년 김은진, 김기범, 황현욱은 바로가기 파일이나 점프리스트 기록을 삭제하는 방법은 잘 알려져 있기 때문에 사용자에 의해 쉽게 삭제될 수 있는 문제점을 지적하고 일반 사용자가 쉽게 접근할 수 없는 파일 MFT 엔트리내 $OBJECT_ID 속성의 존재 여부를 이용하여 사용자가 해당 파일에 접근하였는지 확인할 수 있는 방법을 제안하였다[15]. 2017년 강세림, 김소람, 조재형, 김종성은 최근 사용한 파일이나 폴더의 저장 위치 및 시간 정보, 고정 여부, 접근 횟수, 웹브라우저 방문 기록 등을 기록하는 점프리스트 조사의 중요성을 인식하고 기존 점프리스트 분석 도구들의 문제점을 개선하여 통합적인 점프리스트 분석 도구 개발 방안을 제안하였다[16].

한편 일반적인 파일에 대한 사용 흔적을 조사하는 방법 대신 특정 문서 파일들에 한정하여 적용 가능한 포렌식 방법에 대한 연구들도 있었다. 2009년 서기민, 권혁돈, 방제완, 장기식, 오승진은 파일시스템과 별도로 MS Office 파일 내부에 시간 정보가 저장되고 있음을 밝히고 이러한 시간 정보를 이용하여 문석 작성이나 수정이 이루어진 시간을 파악할 수 있음을 보였다[17]. 2020년 박세율과 이상진은 HWP 파일 구조를 분석하여 문서 개체 생성 시간을 파악할 수 있는 InstanceID를 분석하여 편집 이력을 확인하는 방법을 제안하였다[18]. 2020년 이연주, 김정민, 이성진은 폴라리스 오피스(https://www.polarisoffice.com/ko/office)를 이용하여 문서를 편집할 경우 윈도우 시스템이나 맥 시스템에 남겨지는 아티팩트를 조사하여 사용자가 수행한 작업을 추적할 수 있음을 보였다[19].

그러나 이상의 선행된 연구 결과들이 갖고 있는 한계는 다음과 같다:

- $LogFile에 남아 있는 로그 레코드들을 분석하여 윈도우 시스템에서 일반적인 파일 및 디렉토리의 생성, 삭제, 복사, 이동, 이름변경 행위 흔적을 조사할 수 있는 방법에 관한 연구는 있었으나, 이러한 연구 결과만으로는 특정 어플리케이션이 특정 파일을 오픈하여 열람하고 수정할 때 $LogFile에 어떤 로그 레코드를 남기는지 알 수 없다.
- MS Office를 이용하여 MS Office 문서를 단순 열람하거나 작성(수정)하는 행위는 정상적인 문서 열람 및 작성 행위이기 때문에 안티 포렌식 행위 탐지 기법을 적용하여 MS Office 문서 파일 열람 및 작성 흔적을 조사할 수 없다.
- 바로가기 파일로 구성된 최근사용문서나 점프리스트는 사용자가 관련 정보를 쉽게 삭제할 수 있으며 남아 있다 하더라도 가장 최근 사용 정보만을 가지고 있을 뿐이다;
- 오피스 파일 내부에 기록된 정보의 경우 비교적 사용자가 쉽게 삭제하거나 수정하기 어려운 측면이 있을지는 몰라도 역시 가장 최근의 수정 기록만을 가지고 있을 뿐이다;
- 더욱이 기존 어떤 연구들도 파일 수정 작업으로 변경되는 파일의 크기를 추적할 수 있는 방법을 제안한 연구는 없었다.

Ⅲ. NTFS $LogFile

$LogFile 파일은 NTFS 볼륨에 존재하는 파일로서 MFT 엔트리 번호는 2이며 NTFS 트랜잭션 로그 파일로 시스템

오류나 갑작스런 전원 차단시 작업 중이던 파일을 복구하는데 사용된다. 기본적으로 운영체제가 설치된 볼륨 C:\에만 존재하는 $UsnJrnl 저널과 달리 모든 NTFS 볼륨 마다 존재하며 트랜잭션 작업을 순차적으로 증가하는 고유의 LSN($LogFile Sequence Number) 순서로 기록한다. $LogFile은 파일의 크기를 변경할 수 있긴 하지만 일반적으로 64MB(67,108,864 바이트) 크기를 갖는다. 운영체제가 설치된 볼륨 C:\의 경우는 운영체제에 의한 시스템 작업을 기록하기 때문에 통상 3~4시간 정도의 트랜잭션 기록만을 유지한다. 그러나 운영체제가 설치되지 않은 NTFS 볼륨의 경우는 시스템 작업이 볼륨 C:\에서와 같이 많지 않기 때문에 3일에서 15일 정도로(해당 볼륨에서의 작업량이 많지 않을 경우에는 1달 이상이 될 수도 있음) 비교적 오랜 기간의 로그 레코드 기록이 유지된다.

$LogFile은 4096 바이트 크기의 페이지들로 구성되며 최초 2개 페이지로 구성된 재시작 영역과 실제 트랜잭션들을 기록하는 로깅 영역으로 나뉘며, 로깅 영역은 다시 버퍼 페이지 영역과 일반 페이지 영역으로 나뉜다. 버퍼 페이지 영역은 2개의 페이지로 구성되는데 일반 페이지 영역에 레코드들이 저장되기 전에 임시 저장되는 가장 최근의 레코드들이 저장된다. 각 트랜잭션은 다시 다음의 〈표 1〉와 같은 여러개의 작업 레코드들로 구성된다. 각 작업 레코드는 88바이트의 헤더와 데이터로 구성되는데 데이터는 Redo 데이터와 Undo 데이터로 구성된다. Redo 데이터와 Undo 데이터를 가지고 어떤 작업을 해야 하는지는 헤더 시작으로부터 48 바이트 위치에 있는 2바이트 Redo Opcode와 2바이트 Undo Opcode로 표시된다. 각 작업 코드가 수행하는 작업의 명칭은 〈표 2〉와 같다. 이번 장에서는 $LogFile에 대하여 필요한 최소한의 내용만을 아주 간략하게 소개하였다. $LogFile에 대한 보다 상세한 내용은 참고문헌 [1][2]를 참고하기 바란다.

〈Table 1〉 Structure of Log Records in $LogFile

<table>
<tr><td colspan="16">0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td></tr>
<tr><td colspan="8">This LSN</td><td colspan="8">Previous LSN</td></tr>
<tr><td colspan="8">Client Undo LSN</td><td colspan="4">Client Data Length</td><td colspan="2">Seq. No.</td><td colspan="2">Client ID</td></tr>
<tr><td colspan="4">Record Type</td><td colspan="4">Transaction ID</td><td colspan="2">Flags</td><td colspan="6">Reserved</td></tr>
<tr><td colspan="2">Redo Op.</td><td colspan="2">Undo Op.</td><td colspan="2">RedoOffset</td><td colspan="2">Redo Len.</td><td colspan="2">UndoOffset</td><td colspan="2">Undo Len.</td><td colspan="2">TargetAttr</td><td colspan="2">LCNto Foll</td></tr>
<tr><td colspan="2">RecOffset</td><td colspan="2">AttrOffset</td><td colspan="2">MFT Cluster Idx</td><td colspan="2">Alignment</td><td colspan="4">Target VCN</td><td colspan="4">Alignment</td></tr>
<tr><td colspan="4">Target LCN</td><td colspan="4">Alignment</td><td colspan="8"></td></tr>
<tr><td colspan="16">Redo Data(Variable) + Undo Data(Variable)</td></tr>
</table>

〈Table 2〉 Redo/Undo Opcode

작업코드	작업 내용	작업코드	작업 내용
0x00	Noop	0x0F	AddIndex엔트리Allocation
0x01	CompensationlogRecord	0x12	SetIndex엔트리VenAllocation
0x02	InitializeFileRecordSegment	0x13	UpdateFileNameRoot
0x03	DeallocateFileRecordSegment	0x14	UpdateFileNameAllocation
0x04	WriteEndofFileRecordSegement	0x15	SetBitsInNonresidentBitMap
0x05	CreateAttribute	0x16	ClearBitsInNonresidentBitMap
0x06	DeleteAttribute	0x19	PrepareTransaction
0x07	UpdateResidentValue	0x1A	CommitTransaction
0x08	UpdataeNonResidentValue	0x1B	ForgetTransaction
0x09	UpdateMappingPairs	0x1C	OpenNonresidentAttribute
0x0A	DeleteDirtyClusters	0x1F	DirtyPageTableDump
0x0B	SetNewAttributeSizes	0x20	TransactionTableDump
0x0C	Addindex엔트리Root	x21	UpdateRecordDataRoot
0x0D	Deleteindex엔트리Root		

Ⅳ. $LogFile을 이용한 MS Office 문서 파일 열람 및 작성 과정 분석

이번 장에서는 Word, Powerpoint, Excel와 같은 MS Office 문서를 열람하거나 작성하는 과정에서 NTFS $LogFile이 어떤 로그를 기록하는지 조사하기 위하여 실험 환경을 구축하고 실험 환경 시스템에서 $LogFile에 기록된 MS Office 작업 로그 레코드들을 분석하고자 한다.

4.1 실험 환경 구축

운영체제가 설치된 볼륨 C:\가 아니라 기타 NTFS 볼륨의 $LogFile에 기록된 MS Office 문서 열람 및

작성 흔적을 조사하기 위하여 다음과 같이 실험 환경을 구축하였다.

■ 하드웨어:
- 64비트 인텔 프로세서가 장착된 노트북
- 32GB 외장 SSD: NTFS로 포맷(섹터 크기: 512 바이트, 클러스터 크기: 4,096 바이트)
  외장 SSD는 시스템에 볼륨 F:\로 마운트되었으며, 실질적인 MS Office 문서 열람 및 작성은 모두 32GB 외장 SSD에서 수행되었다.

■ 소프트웨어
- 64비트 한글윈도우10 운영체제
- MS Office 2016: 확장자 docx, pptx, xlsx를 갖는 MS Office 문서 작업을 위하여 사용.

$LogFile에 MS Office 문서 작업이 남기는 흔적을 분석하기 위하여, 외장 SSD(볼륨 F:\) 아래에 MS Office(MFT 엔트리 40) 폴더를 생성하고 MS Office 폴더 아래에 다음과 같은 3개의 MS Office 문서 파일을 복사하였다. 복사된 3개의 파일들의 속성은 다음과 같다.

(1) 파일명: Word_Test.docx(MFT 엔트리 44), 파일 크기: 454,261바이트,
생성/접근시간: 2022.01.04. 15:02:14, 수정시간: 2022.01.03. 15:20:55
(2) 파일명: PowerPoint_Test.pptx(MFT 엔트리 45), 파일 크기: 1,187,245바이트,
생성/접근시간: 2022.01.04. 15:03:25, 수정시간: 2022.01.04. 14:04:21
(3) 파일명: Excel_Test.xlsx(MFT 엔트리 46), 파일 크기: 140,155바이트,
생성/접근시간: 2022.01.04. 15:04:22, 수정시간: 2022.01.04. 11:50:04

■ 예제 파일 Word_Test.docx 문서 열람 및 작성
① 먼저 상기 Word_Test.docx 파일에 대하여 해당 파일을 오픈하고 단순 열람만 한 후 해당 파일을 종료하였다.
② 다시 Word_Test.docx 파일을 오픈한 후 1차로 수정 작업을 한 후 종료없이 중간 저장한 후 다시 2차 수정 작업을 수행하고 종료하면서 저장하였다.

■ 예제 파일 PowerPoint_Test.pptx & Excel_Test.xlsx 문서 열람 및 작성: 상기 Word_Test.docx와 같은 동일한 과정으로 해당 문서 파일을 열람 및 작성 하였다.

다음의 2절과 3절에서는 F:\$LogFile에 남겨진 작업 레코드 기록들을 조사하여 상기 MS Office 문서 열람 및 작성 작업 과정을 재구성할 수 있음을 보이고자 한다.

4.2 $LogFile을 이용한 MS Office 문서 열람 과정 분석

Ⅵ장 1절에서 설명한 방법대로 3개의 MS Office 문서 파일(Word_Test.docx, PowerPoint_Test.pptx, Excel_Test.xlsx)들을 수정이나 저장하는 작업없이 단순 열람만 하였을 경우 F:\ 볼륨내 $LogFile에 어떤 흔적을 남기는지 조사하고 그러한 흔적을 바탕으로 문서 열람 작업을 재구성하고자 한다. $LogFile은 MS Office 문서 파일에 대한 단순 열람만 하더라도 Ⅲ장 〈표 2〉에 기술된 수십개의 작업 레코드들을 기록하기 때문에 지면 관계상 MS Office 파일 열람을 구성하는 모든 작업 레코드들을 기술하기는 어렵다. 따라서, 본 논문에서는 MS Office 폴더내 파일의 추가/삭제와 관련된 작업 레코드(Add/Delete Index Entry Root 또는 Add/Delete Index Entry Allocation)들과 파일의 생성/삭제와 관련된 작업 레코드(Initailize/Deallocate File Record Segment)들을 중심으로 기술하고자 한다. 〈그림 1〉은 Word_Test.docx 문서 파일에 대한 단순 열람 과정을 $LogFile에 기록된 주요 작업 레코드들로 재구성한 것을 도식화한 것이다. 〈그림 1〉에 나타난 작업 과정을 단계별로 설명하면 다음과 같다:

① 시스템 사용자 행위 1: Word_Test.docx 파일을 오픈한다.
② LSN 16,798,811: Word_Test.docx 문서 파일의 부모 디렉토리 MSOffice(MFT 엔트리 40) 아래에 임시파일 ~$rd_Word_Test.docx을 추가하기 위하여 인덱스 엔트리를 추가한다.
③ LSN 16,798,836: 현재 사용되고 있지 않는 MFT 엔트리 47을 ~$rd_Test.docx 파일에 할당하고 초기화한다. 이때, $FILE_NAME 속성내 파일명은 ~$rd_Test.docx로 하고, $STANDARD_INFORMATION 속성의 생성/수정/접근시간은 MFT 엔트리 47이 할당된 시간으로 초

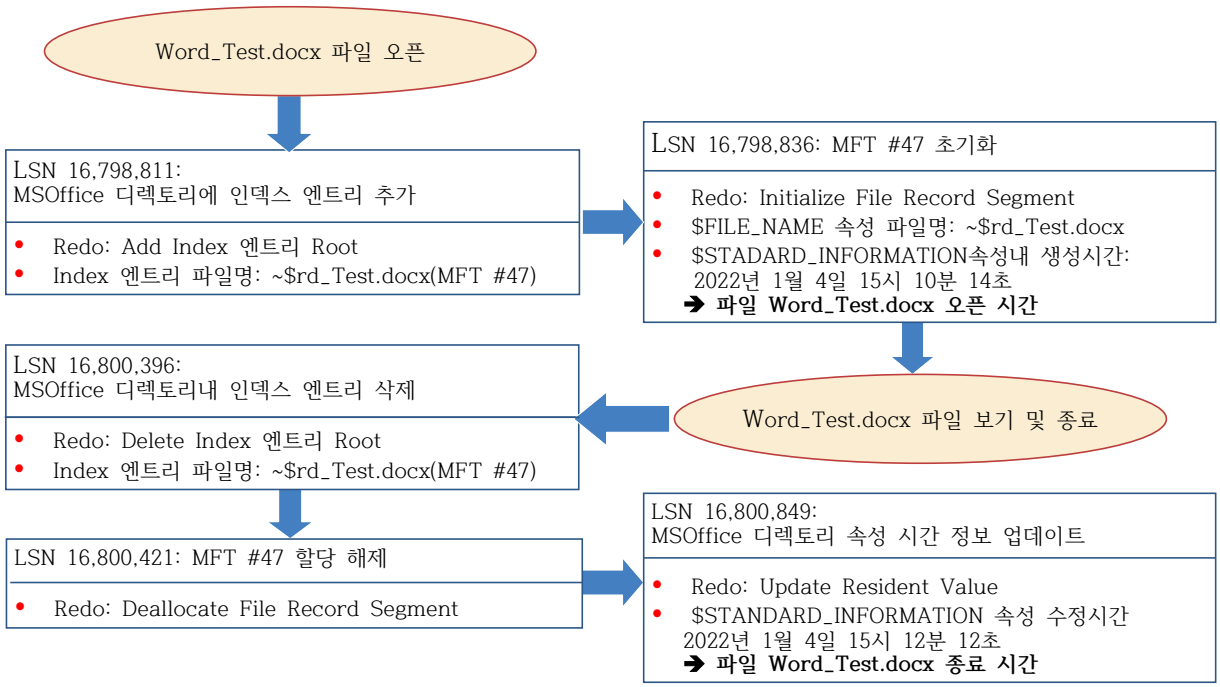
기화한다. 따라서, 해당 시간이 파일을 오픈한 시간이 된다.

④ 시스템 사용자 행위 2: 사용자가 문서 수정 및 저장없이 해당 문서 파일을 단순 열람한 후 종료한다.

⑤ LSN 16,800,396: 임시파일 ~$rd_Test.docx를 삭제하기 위해 부모 디렉토리 MSOffic내 해당 파일의 인덱스 엔트리를 삭제한다.

⑥ LSN 16,800,421: MFT 엔트리 47을 할당 해제한다.

⑦ LSN 16,800,849: MSOffice 디렉토리(MFT 엔트리 40)의 $STANDARD_INFORMATION 속성 시간 정보를 업데이트 한다. LSN 16,800,396, 16,800,421에는 삭제 시간을 참고할 시간 정보가 없기 때문에, MS Office 디렉토리의 MFT 엔트리 40의 $STANDARD_INFORMATION 속성 수정시간이 사용자가 Word_Test.docx 파일 보기를 종료한 시간이 된다.



〈Figure 1〉 Reading Process of Word_Test.docx

상기 예에서는 MSOffice 폴더 아래에 파일이 3개 밖에 없으나 파일 또는 디렉토리가 많을 경우 Add Index Entry Root/Delete Index Entry Root 대신 Add Index Entry Allocation/Delete Index Entry Allocation이 올 수 있다.

PowerPoint_Test.pptx, Excel_Test.xlsx 문서 파일에 대한 저장없는 단순 열람도 상기 Word_Test.docx 문서 파일 열람과 동일한 작업 과정으로 구성되는데, 해당 파일을 오픈할 때 생성되는 임시 파일만이 PowerPoint_Test.pptx의 경우는 ~$PowerPoint_Test.pptx, Excel_Test.xlsx의 경우에는 ~$Excel_Test.xlsx로 바뀔 뿐이다.

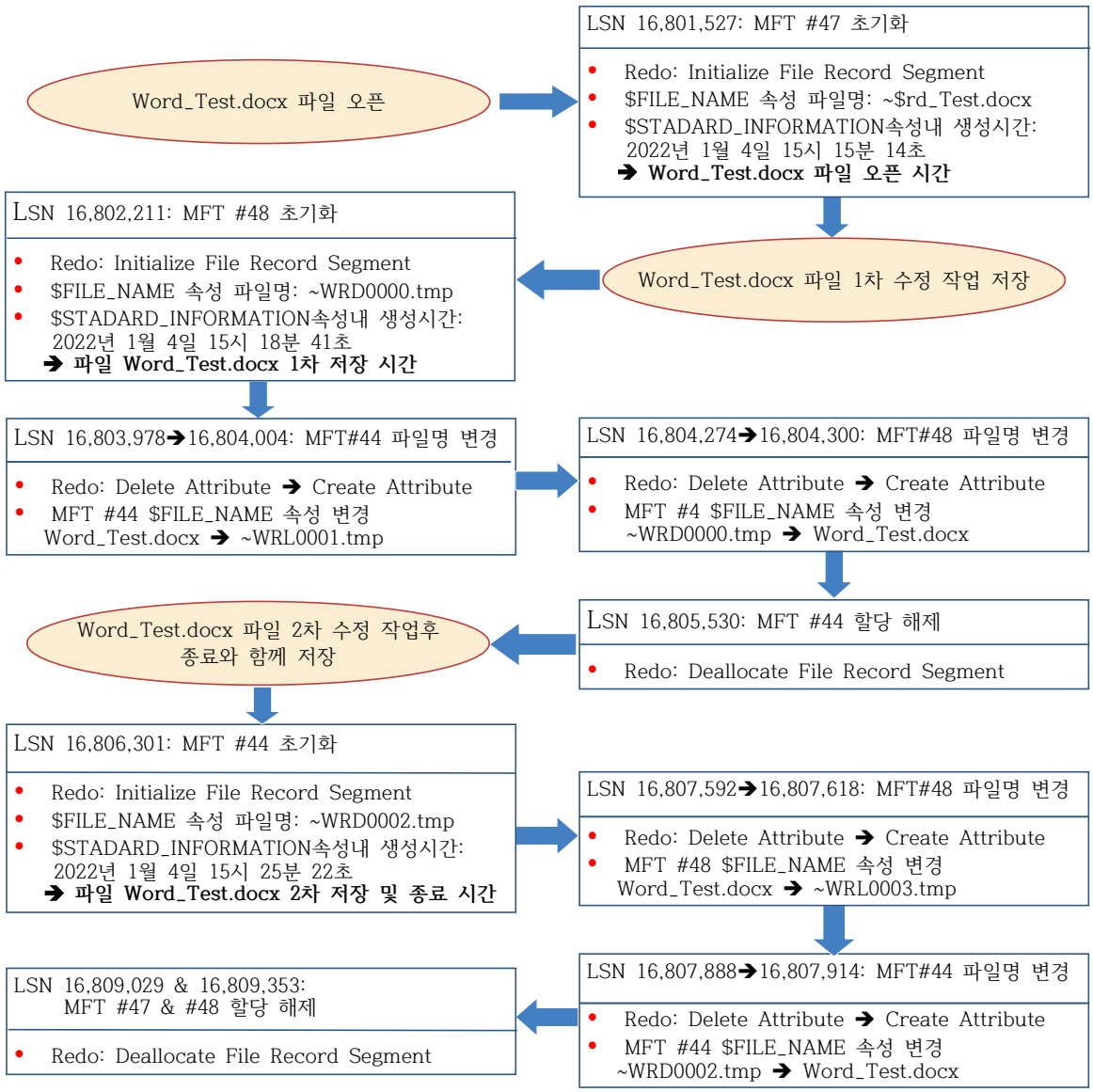
4.3 $LogFile을 이용한 MS Office 문서 작성 작업 과정 분석

이번 절에서는 $LogFile에 남아 있는 기록을 이용하여 사용자의 MS Office 문서 작성 작업 과정을 재구성하고자 한다. MS Office 작성 작업 과정은 수백개의 작업 레코드들로 구성되기 때문에 표현하기 어려운 점이 있어 본 절에서는 'Initialize/Deallocate File Record Segment', 'Delete Attribute/Create Attribute' 작업 레코드들을 중심으로 문서 작성 과정을 재구성하고자 한다.

4.3.1 MS Word 문서 작성 작업 과정 재구성

〈그림 2〉는 1절에서 소개한 F:\MSOffice에 있는 예제 파일 Word_Test.docx에 대한 2차에 걸친 파일 수정 작업 과정을 재구성한 것이다. 〈그림 2〉에 나타난 작업 과정을 단계별로 설명하면 다음과 같다:

① 시스템 사용자 행위 1: Word_Test.docx 파일을 오픈한다.
② LSN 16,801,527: 이 작업에 앞서 MSOffice 디렉토리(MFT 엔트리 40)에 파일 ~$rd_Test.docx의 인덱스 엔트리를 추가하는 작업이 선행된다(LSN 16,801,502). 현재 사용되고 있지 않는 MFT 엔트리 47을 ~$rd_Test.docx 파일에 할당하고 초기화한다. MFT 엔트리 47 $STANDARD_INFORMATION 속성의 생성/수정/접근시간은 해당 파일 오픈 시간이 된다.
③ 시스템 사용자 행위 2: 사용자는 1차적으로 해당 문서 파일에 대한 수정 작업을 한 후 종료하지 않고 저장만 한다.
④ LSN 16,802,211: 이 작업에 앞서 MSOffice 디렉토리(MFT 엔트리 40)에 파일 ~WRD0000.tmp의 인덱스 엔트리를 추가하는 작업이 선행된다(LSN 16,802,186). ~WRD0000.tmp에는 사용자의 Word_Test.docx 파일에 대한 1차 수정 작업 내용이 저장되어 있다. 현재 사용되고 있지 않는 MFT 엔트리 48을 ~WRD0000.tmp 파일에 할당하고 초기화한다. MFT 엔트리 48 $STANDARD_INFORMATION 속성의 생성/수정/접근시간은 파일 Word_Test.docx 1차 저장 시간으로 볼 수 있다(정확한 시간과는 1초 이내로 차이날 수 있으나 포렌식 수사 관점에서 큰 의미는 없다).



〈Figure 2〉 Writing Process of Word_Test.docx

⑤ LSN 16,803,978➔16,804,004: MSOffice 디렉토리(MFT 엔트리 40)에서 파일 Word_Test.docx의 인덱스 엔트리를 삭제하는 작업이 선행된다(LSN 16,803,953), MFT 엔트리 44의 $FILE_NAME 속성(파일명: Word_Test.docx)을 삭제한 후(LSN 16,803,978: Delete Attribute), $FILE_NAME 속

성(파일명: ~WRL0001.tmp)을 생성하여 추가한다(LSN 16,804,004: Create Attribute). 이후, MSOffice 디렉토리(MFT 엔트리 40)에 파일 ~WRL0001.tmp의 인덱스 엔트리를 추가하는 작업으로 마무리된다(LSN 16,804,030),

⑥ LSN 16,804,274➔16,804,300: MSOffice 디렉토리(MFT 엔트리 40)에서 파일 ~WRD0000.tmp의 인덱스 엔트리를 삭제하는 작업이 선행된다(LSN 16,804,249), MFT 엔트리 48의 $FILE_NAME 속성(파일명: ~WRD0000.tmp)을 삭제한 후(LSN 16,804,274: Delete Attribute), $FILE_NAME 속성(파일명: Word_Test.docx)을 생성하여 추가한다(LSN 16,804,300: Create Attribute). 이후, MSOffice 디렉토리(MFT 엔트리 40)에 파일 Word_Test.docx의 인덱스 엔트리를 추가하는 작업으로 마무리된다(LSN 16,804,326),

⑦ LSN 16,805,530: MSOffice 디렉토리(MFT 엔트리 40)에서 파일 ~WRL0001.tmp의 인덱스 엔트리를 삭제하는 작업이 선행된 후(LSN 16,805,483), MFT 엔트리 44를 할당 해제한다. 이 시점에서 Word_Test.docx 파일의 MFT 엔트리는 최초 44에서 48로 변경된다.

⑧ 시스템 사용자 행위 3: 사용자가 2차적으로 해당 문서 파일에 대한 수정 작업을 하고 종료하면서 저장한다.

⑨ LSN 16,806,301: 이 작업에 앞서 MSOffice 디렉토리(MFT 엔트리 40)에 파일 ~WRD0002.tmp의 인덱스 엔트리를 추가하는 작업이 선행된다(LSN 16,806,276). ~WRD0002.tmp에는 사용자의 Word_Test.docx 파일에 대한 2차 수정 작업 내용이 저장되어 있다. 현재 사용되고 있지 않는 MFT 엔트리 44를 ~WRD0002.tmp 파일에 할당하고 초기화한다. MFT 엔트리 44 $STANDARD_INFORMATION 속성의 생성/수정/접근 시간은 파일 Word_Test.docx 2차 저장 및 종료 시간으로 볼 수 있다. 보다 정확한 파일 저장 시간을 얻기 위해서는 MFT 엔트리 44의 $STANDARD_INFORMATION 속성내 시간 정보를 업데이트하는 후속적인 작업 레코드 LSN 16,807,330(Redo: Update Resident Value)을 분석하여야 하지만 시간 차이는 1초 이내이다.

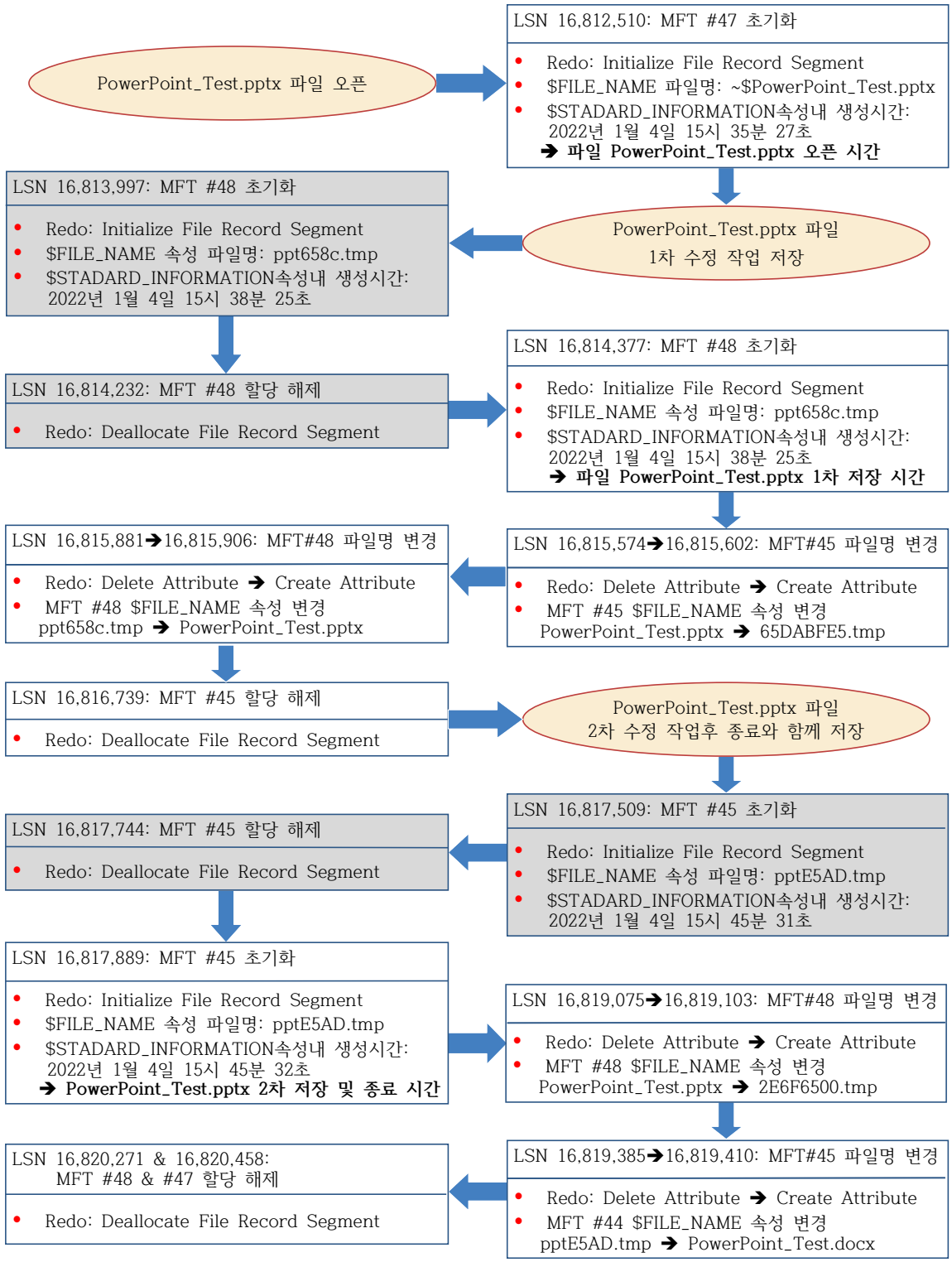
⑩ LSN 16,807,592➔16,807,618: MSOffice 디렉토리(MFT 엔트리 40)에서 파일 Word_Test.docx의 인덱스 엔트리를 삭제하는 작업이 선행된다(LSN 16,807,567), MFT 엔트리 48의 $FILE_NAME 속성(파일명: Word_Test.docx)을 삭제한 후(LSN 16,807,592: Delete Attribute), $FILE_NAME 속성(파일명: ~WRL0003.tmp)을 생성하여 추가한다(LSN 16,807,618: Create Attribute). 이후, MSOffice 디렉토리(MFT 엔트리 40)에 파일 ~WRL0003.tmp의 인덱스 엔트리를 추가하는 작업으로 마무리된다(LSN 16,807,644),

⑪ LSN 16,807,888➔16,807,914: MSOffice 디렉토리(MFT 엔트리 40)에서 파일 ~WRD0002.tmp의 인덱스 엔트리를 삭제하는 작업이 선행된다(LSN 16,807,863), MFT 엔트리 44의 $FILE_NAME 속성(파일명: ~WRD0002.tmp)을 삭제한 후(LSN 16,807,888: Delete Attribute), $FILE_NAME 속성(파일명: Word_Test.docx)을 생성하여 추가한다(LSN 16,807,914: Create Attribute). 이후, MSOffice 디렉토리(MFT 엔트리 40)에 파일 Word_Test.docx의 인덱스 엔트리를 추가하는 작업으로 마무리된다(LSN 16,807,948),

⑫ LSN 16,809,029 & 16,809,353: MSOffice 디렉토리(MFT 엔트리 40)에서 파일 ~$rd_Test.docx의 인덱스 엔트리를 삭제한 후(LSN 16,809,004), MFT 엔트리 47를 할당 해제한다(LSN 16,809,029). MSOffice 디렉토리(MFT 엔트리 40)에서 파일 ~WRL0003.tmp의 인덱스 엔트리를 삭제한 후(LSN 16,809,306), MFT 엔트리 48을 할당 해제한다(LSN 16,809,353).

상기 문서 작성에서는 2차 수정 작업 이후 Word_Test.docx 파일을 종료와 함께 저장하였기 때문에 MFT 엔트리 47(~$rd_Test.docx)와 MFT 엔트리 48(~WRL0003.tmp)할당 해제가 동시에 일어났다. 만일 2차 수정 작업을 종료와 함께 저장하지 않고 저장을 먼저하고 난 이후에 뒤이어 파일을 종료할 경우에는 LSN 16,806,301이 파일 종료 시간이 될 수 없고 MFT 엔트리 47(~$rd_Test.docx) 할당 해제 시간을 〈그림 1〉에서와 같은 방법으로 별도로 계산하여야 한다(LSN 16,809,029 이후 MS Office 디렉토리인 MFT 엔트리 40의 $STANDARD_INFORMATION 속성 수정시간을 업데이트하는 로그 레코드(Redo: Update Resident Value)로부터 얻을 수 있다).

4.3.2 MS Powerpoint 문서 작성 작업 과정 재구성



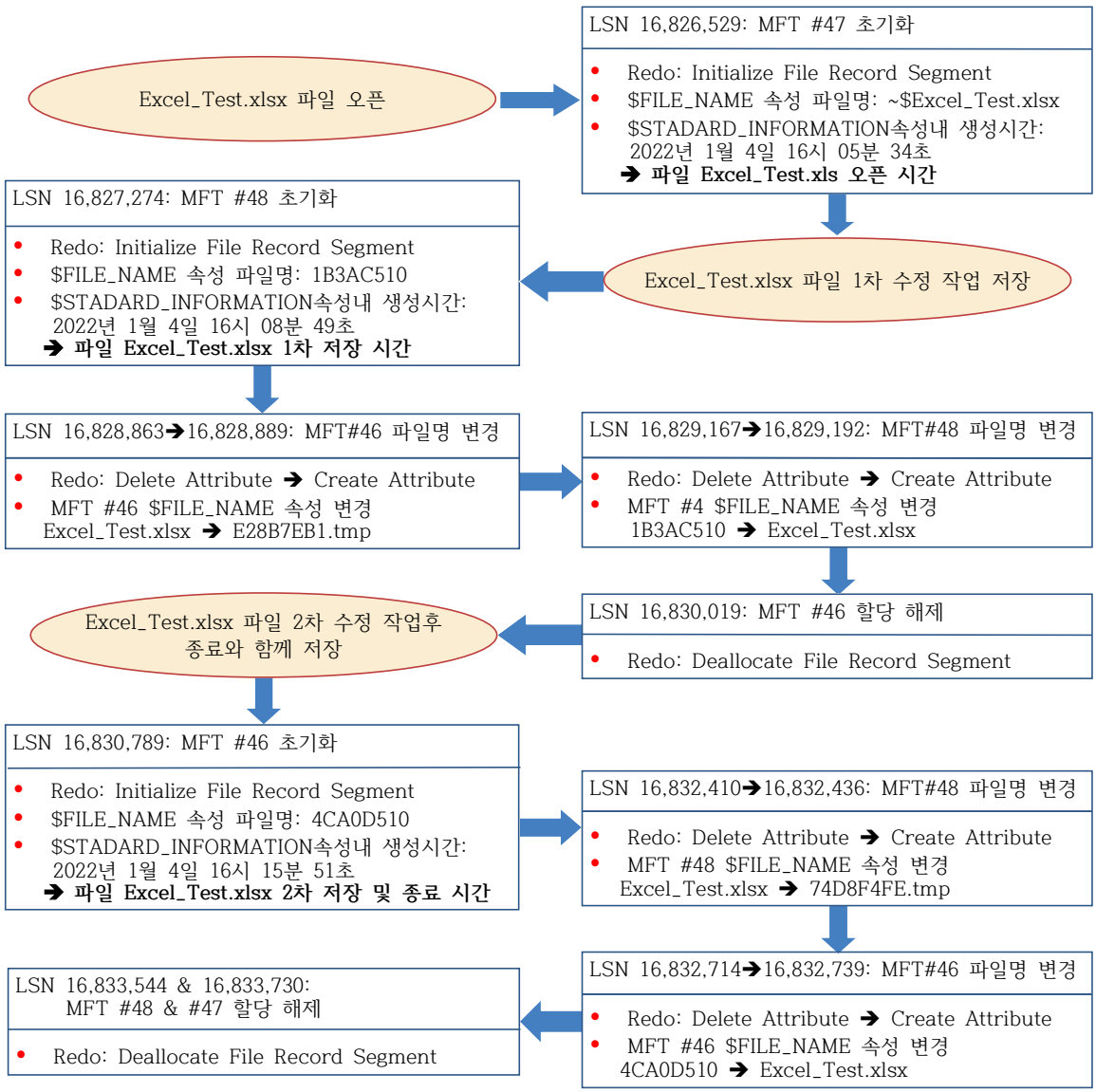
〈Figure 3〉 Writing Process of PowerPoint_Test.pptx

〈그림 3〉은 1절에서 소개한 F:\MSOffice에 있는 예제 파일 PowerPoint_Test.pptx에 대한 2차에 걸친 수정 작업 과정을 F:\$LogFile에서 'Initialize/Deallocate File Record Segment', 'Create/Delete Attribute' 작업 레코드들을 중심으로 재구성한 것이다. 자세한 단계별 작업 과정은 MS Word 작업 과정에 대한 설명을 참고하여 이해하기 바란다. MS Word의 경우와 비교하였을 때, 주요한 차이점은 MS Powerpoint 문서 파일 수정 과정에서 생성되는 임시파일명들의 형태가 다르다는 것이다. 그리고 특이하게도 무슨 이유인지 몰라도 MS Powerpoint 문서 파일 수정 과정에서는 임시파일 ppt658c.tmp와

pptE5AD.tmp을 생성하고(LSN 16,813,997 & 16,817,509) 삭제한 후(LSN 16,814,232 & 16,817,744) 다시 생성하는(LSN 16,814,377 & 16,817,889) 과정을 반복한다는 것이다.

MS Word의 경우와 마찬가지로, 만일 2차 수정 작업을 종료와 함께 저장하지 않고 저장을 먼저하고 난 이후에 뒤이어 파일을 종료할 경우에는 LSN 16,817,889가 파일 종료 시간이 될 수 없고 MFT 엔트리 47(~$PowerPoint_Test.pptx) 할당 해제 시간을 〈그림 1〉에서와 같이 해당 파일의 부모 디렉토리(MS Office)의 $STANDARD_INFORMATION 속성내 수정 시간 업데이트와 관련된 로그 레코드로부터 별도로 계산하여야 한다.

4.3.3 MS Excel 문서 작성 작업 과정 재구성



〈Figure 4〉 Writing Process of Excel_Test.xlsx

〈그림 4〉는 1절에서 소개한 F:\MSOffice에 있는 예제 파일 Excel_Test.xlsx에 대한 2차에 걸친 수정 작업 과정을 F:\$LogFile에서 'Initialize/Deallocate File Record Segment', 'Create/Delete Attribute' 작업 레코드들을 중심으로 재구성한 것이다. Excel_Test.xlsx을 수정하였을 때 생성된 임시파일들은 E28B7EB1.tmp, 1B3AC510, 74D8F4FE.tmp, 4CA0D510과 같은 형태로서 MS Word 그리고 MS Powerpoint의 경우와도 차이가 있다. 그리고, Powerpoint 수정의 경우와 같이 임시파일이 생성되고 삭제되고 다시 생성되는 특이한 현상은 없었으며 전반적으로 임시파일명의 형태를 제외하면 MS Word와 동일한 과정으로 진행된다. 자세한 단계별 작업 과정은 MS Word에 대한 설명을 참고하여 이해하기 바란다.

2차 수정 작업을 종료와 함께 저장하지 않고 저장을 먼저하고 난 이후에 뒤이어 파일을 종료할 경우에는

LSN 16,830,789가 파일 종료 시간이 될 수 없고 MFT 엔트리 47(~$Excel_Test.pptx) 할당 해제 시간을 〈그림 1〉에서와 같이 해당 파일의 부모 디렉토리(MS Office)의 $STANDARD_INFORMATION 속성 내 수정 시간 업데이트와 관련된 로그 레코드로부터 별도로 계산하여야 한다.

4.4 MS Office 파일 열람 및 작성 작업 과정 요약

앞의 2절과 3절의 결과는 비록 특정 실험 환경에서 주어진 3개의 예제 파일들(Word_Test.docx, PowerPoint_Test.ppt, Excel_Test.xlsx)에 대한 결과이지만, 저장매체 SSD와 HDD의 NTFS 볼륨에 대한 여러 차례의 실험을 통하여 〈표 3〉과 〈그림 5〉와 같은 결과를 얻을 수 있다.

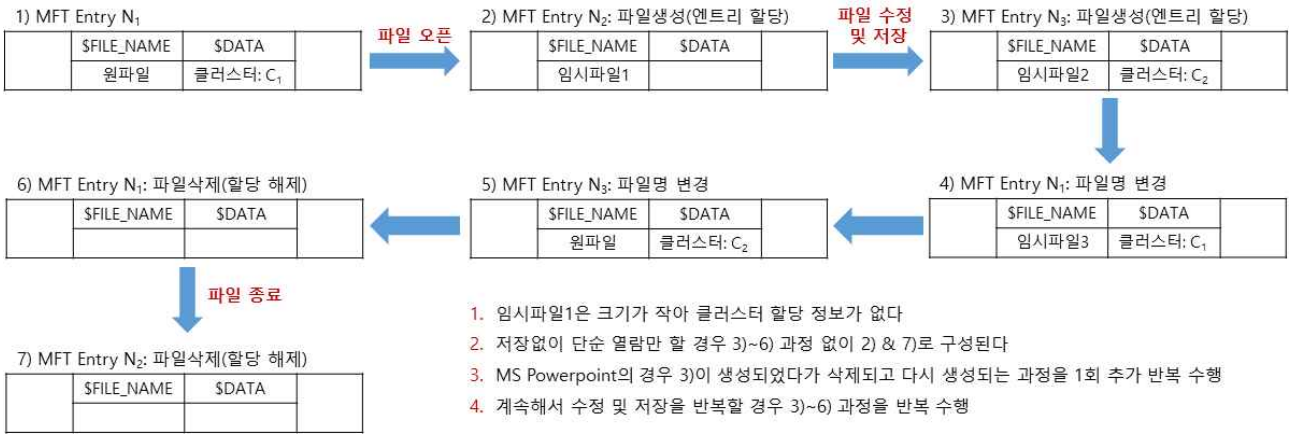
MS Office 문서파일을 단순 열람만 할 경우에는 〈표 3〉과 같은 임시파일1만 생성되었다가 삭제되고, MS Office 문서 파일을 수정이나 저장할 경우에는 임시파일1과 더불어 임시파일2&3이 생성된다는 사실을 알 수 있다. 〈그림 3〉에서 보았듯이 MS Powerpoint 문서 파일의 경우에는 무슨 이유인지 모르겠지만 임시파일2가 생성되었다가 삭제되고 다시 생성되는 과정을 1회 반복한다. MS Word 문서 파일은 파일을 열어 어떤 수정도 없이 저장하는 경우 단순 열람하는 경우와 동일하게 임시파일1만 생성되고 임시파일2&3은 생성되지 않는다. 반면, MS Powerpoint와 MS Excel 문서 파일은 파일을 열어 어떤 수정도 하지 않은 상태에서 저장하여도, 수정하여 저장하는 경우와 동일하게 처리되며(따라서 임시파일1,2&3 모두 생성) 클러스터 할당 정보, 파일 크기, 파일 수정시간, 파일 접근 시간 등도 역시 변경된다는 것을 확인할 수 있다. 이때 해당 파일의 크기는 수백 바이트 정도 수준에서 변경된다.

결론적으로 우리는 〈표 3〉과 〈그림 5〉에서 볼 수 있는 임시파일1,2,3와 같은 고유한 형태의 파일들이 생성되고 삭제되는 흔적들을 $LogFile에서 조사함으로써 언제 해당 MS Office 문서 파일이 열람되고 작성(수정)되었는지 추적할 수 있다.

저장매체가 SSD 또는 HDD가 아닌 USB 메모리인 경우 MS Powerpoint와 MS Excel 문서 파일은 SSD, HDD와 동일하나 MS Word 문서 파일의 경우는 차이가 있다. 관련된 보다 자세한 내용은 Ⅶ장을 참조하기 바란다.

〈Table 3〉 Temporary Files Showing MS Office File Reading and Writing

	원파일	열람및작성흔적	작성흔적(단순 열람시 생성 안됨)	
		임시파일1	임시파일2	임시파일3
MS Word	*.docx	원파일명 8글자 이상: 앞의 2글자 '~$'로 대치 원파일명 7글자: 앞의 1글자를 '~$'로 대치 원파일명 6글자 이하: 앞에 '~$' 추가	~WRDXXXX.tmp	~WRLYYYY.tmp
MS Powerpoint	*.pptx	앞에 '~$' 추가 (~$*.pptx)	pptXXXX.tmp	YYYYYYYY.tmp
MS Excel	*.xlsx	앞에 '~$' 추가 (~$*.xlsx)	XXXXXXXX	YYYYYYYY.tmp
설 명	원파일의 내용은 파일 수정 및 저장 마지막 단계에서 삭제된다 확장자 doc, ppt, xls인 MS Office 파일의 경우 전반적인 과정에 대한 추가 확인 필요	임시파일1이 생성되는 시간은 원파일이 오픈되는 시간 임시파일1이 삭제되는 시간은 원파일의 열람과 수정이 종료되는 시간 저장없이 단순 열람만 할 경우 임시파일2&3 생성 및 삭제 과정없이 임시파일1만 생성되었다가 삭제	XXXX: 서로 다른 4자리 16진수 의미 XXXXXXXX: 서로 다른 8자리 16진수 의미 파일 변경 내용을 저장하기 위해 생성되는 파일로 저장 마무리 단계에서 원파일명으로 변경됨 (임시파일2➔원파일) 임시파일2가 생성되는 시간은 변경 작업이 저장되는 시간	YYYY: 서로 다른 4자리 16진수 의미 YYYYYYYY: 서로 다른 8자리 16진수 의미 수정 및 저장 이전 원파일 내용을 삭제하기 전에 임시로 저장하기 위해 원파일명을 임시파일3로 변경(원파일➔임시파일3) 임시파일2가 원파일명으로 변경 완료되면 삭제됨



〈Figure 5〉 Reading and Writing Process of MS Office Files

Ⅴ. 포렌식 도구를 이용한 MS Office 문서 열람 및 작성 흔적 조사

이번 장에서는 $LogFile 분석 도구 중 하나인 NTFS Log Tracker v.1.7을 사용하여 MS Office 문서 파일 열람 및 작성 흔적을 조사한 결과를 소개하고자 한다[14]. $LogFile 분석 도구에는 NTFS Log Tracker 이외에도 logfileparse.py[20], Joakim Schicht가 개발한 LogFileParser[21]을 포함한 여러 가지 도구들이 있지만 NTFS Log Tracker는 $LogFile에 기록된 수많은 작업 레코드들 중에서 수사관들이 주목해서 보아야 할 이벤트들만을 추출하여 직관적으로 이해하기 쉽도록 사용자에게 보여 주기 때문에 $LogFile을 최초 분석 할 때 아주 유용하다. 본 논문에서는 시각적인 인식을 높이기 위하여 원래의 NTFS Log Tracker 결과 중 몇 개 항목을 삭제하거나 조정하여 표기하였다(EventTime의 경우 모든 이벤트 발생일이 2022년 1월 4일로 동일하여 시간 정보만을 표기하였다).

5.1 MS Word 문서 열람 및 작성 흔적 조사

LSN	EventTime (UTC+9)	Event	Detail	Full Path	Redo	Target VCN	Cluster Index
16798836	15:10:14	File Creation		₩MSOffice₩~$rd_Test.docx	Initialize File Record Segment	0xB	6
16799027		Writing Content of Resident File	Writing Size : 54	₩MSOffice₩~$rd_Test.docx	Update Resident Value	0xB	6
16800421	15:12:12	File Deletion		₩MSOffice₩~$rd_Test.docx	Deallocate File Record Segment	0xB	6
16801527	15:15:14	File Creation		₩MSOffice₩~$rd_Test.docx	Initialize File Record Segment	0xB	6
16801718		Writing Content of Resident File	Writing Size : 54	₩MSOffice₩~$rd_Test.docx	Update Resident Value	0xB	6
16802211	15:18:48	File Creation(File System Tunneling)		₩MSOffice₩~WRD0000.tmp	Initialize File Record Segment	0xC	0
16802853		Writing Content of Non-Resident	Data Runs(in Volume) : 6778(224)	₩MSOffice₩~WRD0000.tmp	Update Mapping Pairs	0xC	0
16803311		Writing Content of Non-Resident	Data Runs(in Volume) : 6778(86)	₩MSOffice₩~WRD0000.tmp	Update Mapping Pairs	0xC	0
16805530	15:18:48	File Deletion		₩MSOffice₩~WRL0001.tmp	Deallocate File Record Segment	0xB	0
16806301	15:25:23	File Creation(File System Tunneling)		₩MSOffice₩~WRD0002.tmp	Initialize File Record Segment	0xB	0
16806555		Writing Content of Non-Resident	Data Runs(in Volume) : 6864(448)	₩MSOffice₩~WRD0002.tmp	Update Mapping Pairs	0xB	0
16809029		File Deletion		₩MSOffice₩~$rd_Test.docx	Deallocate File Record Segment	0xB	6
16809186				₩MSOffice₩Word_Test.docx	Update Resident Value	0xB	0
16809353	15:25:23	File Deletion		₩MSOffice₩~WRL0003.tmp	Deallocate File Record Segment	0xC	0

〈Figure 6〉 Reading and Writing Traces of Word_Test.docx

〈그림 6〉은 예제 파일 F:\MSOffice\Word_Test.docx 열람 및 작성 흔적에 대한 NTFS Log Tracker 1.7 실행 결과이다. NTFS Log Tracker 1.7 실행 결과는 Ⅳ장 3.1절에서 기술한 Word_Test.docx 문서 열람 및 문서 작성 과정에 대한 모든 세부 과정을 반영하고 있지는 않지만 어떤 작업이 있었는지는 충분히 확인할 수 있다.

파일 ~$rd_Test.docx의 생성을 나타내는 LSN 16,798,836(Redo: Initialize File Record Segment)과 추가적인 작업없이 해당 파일의 삭제를 나타내는 LSN 16,800,421(Redo: Deallocate File Record Segment)는 파일 Word_Test.docx에 대한 단순 열람 흔적으로 나타낸다. 파일 ~$rd_Test.docx의 생성을 나타내는 LSN 16,801,527(Redo: Initialize File Record Segment)는 파일 Word_Test.docx를 다시 오픈하였음을 나타낸다. 이후 ~WRD0000.tmp 파일 생성을 나타내는 LSN 16,802,211(Redo: Initialize File Record Segment), ~WRL0001.tmp 파일 삭제를 나타내는 LSN

16,805,530(Redo: Deallocate File Record Segment)는 파일 Word_Test.docx에 대한 1차 수정 작업을 종료없이 저장했음을 의미하며, ~WRD0002.tmp 파일 생성을 나타내는 LSN 16,806,301(Redo: Initialize File Record Segment), ~$rd_Test.docx, ~WRL0003.tmp 파일 삭제를 나타내는 LSN 16,809,029와 16,809,353(Redo: Deallocate File Record Segment)는 2차 수정 작업 후 파일을 종료하면서 저장하였음을 의미한다. ~$rd_Test.docx 파일이 먼저 삭제된 것은 종료와 함께 2차 수정 작업을 저장하였음을 나타낸다. 만일 수정 작업을 저장한 이후에 종료하게되면 ~$rd_Test.docx 파일이 나중에 삭제되며 Word_Test.docx 종료 시간도 달라질 것이다.

다음의 2절과 3절에 소개되는 MS Powerpoint, MS Excel에 대한 실행 결과와 달리 MS Word 실행 결과에서는 임시파일들 ~WRD0000.tmp, ~WRL0001.tmp, ~WRD0002.tmp, ~WRL0003.tmp와 원래 파일 Word_Test.docx의 관계를 보여 주는 결과가 없기 때문에 이러한 파일들간의 보다 정확한 관계를 확인하기 위해서는 작업 레코드에 대한 추가 분석이 필요할 수 있다.

5.2 MS Powerpoint 문서 열람 및 작성 조사

LSN	EventTime (UTC+9)	Event	Detail	Full Path	Redo	Target VCN	Cluster Index
16810506	15:30:25	File Creation		₩MSOffice₩~$PowerPoint_Test.pptx	Initialize File Record Segment	0xB	6
16810882		Writing Content of Resident File	Writing Size : 55	₩MSOffice₩~$PowerPoint_Test.pptx	Update Resident Value	0xB	6
16811449	15:32:14	File Deletion		₩MSOffice₩~$PowerPoint_Test.pptx	Deallocate File Record Segment	0xB	6
16812510	15:35:27	File Creation		₩MSOffice₩~$PowerPoint_Test.pptx	Initialize File Record Segment	0xB	6
16812845		Writing Content of Resident File	Writing Size : 55	₩MSOffice₩~$PowerPoint_Test.pptx	Update Resident Value	0xB	6
16813997	15:38:25	File Creation		₩MSOffice₩ppt658C.tmp	Initialize File Record Segment	0xC	0
16814232		File Deletion		₩MSOffice₩ppt658C.tmp	Deallocate File Record Segment	0xC	0
16814377	15:38:25	File Creation		₩MSOffice₩ppt658C.tmp	Initialize File Record Segment	0xC	0
16814570		Writing Content of Non-Resident	Data Runs(in Volume) : 7135(496)	₩MSOffice₩ppt658C.tmp	Update Mapping Pairs	0xC	0
16815602	15:38:25	Renaming File	PowerPoint_Test.pptx -> 65DABFE5.tm	₩MSOffice₩65DABFE5.tmp	Create Attribute	0xB	2
16815906	15:38:25	Renaming File	ppt658C.tmp -> PowerPoint_Test.pptx	₩MSOffice₩PowerPoint_Test.pptx	Create Attribute	0xC	0
16816739	15:38:25	File Deletion		₩MSOffice₩65DABFE5.tmp	Deallocate File Record Segment	0xB	2
16817509	15:45:31	File Creation		₩MSOffice₩pptE5AD.tmp	Initialize File Record Segment	0xB	2
16817744		File Deletion		₩MSOffice₩pptE5AD.tmp	Deallocate File Record Segment	0xB	2
16817889	15:45:32	File Creation		₩MSOffice₩pptE5AD.tmp	Initialize File Record Segment	0xB	2
16818082		Writing Content of Non-Resident	Data Runs(in Volume) : 7447(576)	₩MSOffice₩pptE5AD.tmp	Update Mapping Pairs	0xB	2
16819103	15:45:32	Renaming File	PowerPoint_Test.pptx -> 2E6F6500.tmp	₩MSOffice₩2E6F6500.tmp	Create Attribute	0xC	0
16819410	15:45:32	Renaming File	pptE5AD.tmp -> PowerPoint_Test.pptx	₩MSOffice₩PowerPoint_Test.pptx	Create Attribute	0xB	2
16820202				₩MSOffice₩PowerPoint_Test.pptx	Update Resident Value	0xB	2
16820271	15:45:32	File Deletion		₩MSOffice₩2E6F6500.tmp	Deallocate File Record Segment	0xC	0
16820458	15:45:32	File Deletion		₩MSOffice₩~$PowerPoint_Test.pptx	Deallocate File Record Segment	0xB	6

〈Figure 7〉 Reading and Writing Traces of PowerPoint_Test.pptx

〈그림 7〉은 예제 파일 F:\MSOffice\PowerPoint_Test.pptx 문서 열람 및 작성 흔적에 대한 NTFS Log Tracker 1.7 실행 결과이다. 파일 ~$PowerPoint_Test.pptx의 생성을 나타내는 LSN 16,810,506(Redo: Initialize File Record Segment)과 추가적인 작업없이 해당 파일의 삭제를 나타내는 LSN 16,811,449(Redo: Deallocate File Record Segment)는 파일 PowerPoint_Test.pptx에 대한 저장없는 단순 열람 흔적을 나타낸다. 파일 ~$PowerPoint_Test.pptx의 생성을 나타내는 LSN 16,812,510(Redo: Initialize File Record Segment)는 파일 PowerPoint_Test.pptx를 다시 오픈하였음을 나타낸다. 파일 ppt658c.tmp을 생성하였다가 삭제하고 다시 생성하였음을 나타내는 LSN 16,813,997(Redo: Initialize File Record Segment), 16,814,232(Redo: Deallocate File Record Segment), 16,814,377(Redo: Initialize File Record Segment), 그리고 파일 65DABFE5.tmp 삭제를 나타내는 LSN 16,816,739(Redo: Deallocate File Record Segment)는 파일 PowerPoint_Test.pptx에 대한 1차 수정 작업의 저장을 나타낸다. 다시 파일 pptE5AD.tmp을 생성하였다가 삭제하고 다시 생성하였음을 나타내는 LSN 16,817,509(Redo: Initialize File Record Segment), 16,817,744(Redo: Deallocate File Record Segment), 16,817,889(Redo: Initialize File Record Segment), 그리고 파일 2E6F6500.tmp과 ~$PowerPoint_Test.pptx 삭제를 나타내는 LSN 16,820,271과 16,820,458(Redo: Deallocate File Record Segment)는 파일 PowerPoint_Test.pptx에 대한 2차 수정 작업을 저장 및 종료하였음을을 나타낸다.

MS Word에 대한 결과와 달리 MS Powerpoint에 대한 결과에서는 중간에 생성되는 임시파일들 65DABFE5.tmp(또는 2E6F6500.tmp), ppt658c.tmp(또는 pptE5AD.tmp)과 원래의 파일 PowerPoint_Test.pptx의 연관성을 보여주는 LSN 16,815,602(또는 16,819,103), 16,815,906(또는

16,819,410)을 확인할 수 있다.

5.3 MS Excel 문서 열람 및 작성 흔적 조사

16824659	16:00:54	File Creation		₩MSOffice₩~$Excel_Test.xlsx	Initialize File Record Segment	0xB	6
16824990		Writing Content of Resident File	Writing Size : 55	₩MSOffice₩~$Excel_Test.xlsx	Update Resident Value	0xB	6
16825384	16:02:48	File Deletion		₩MSOffice₩~$Excel_Test.xlsx	Deallocate File Record Segment	0xB	6
16826529	16:05:34	File Creation		₩MSOffice₩~$Excel_Test.xlsx	Initialize File Record Segment	0xB	6
16826856		Writing Content of Resident File	Writing Size : 55	₩MSOffice₩~$Excel_Test.xlsx	Update Resident Value	0xB	6
16827274	16:08:49	File Creation		₩MSOffice₩1B3AC510	Initialize File Record Segment	0xC	0
16827474		Writing Content of Non-Resident	Data Runs(in Volume) : 7135(96)	₩MSOffice₩1B3AC510	Update Mapping Pairs	0xC	0
16828889	16:08:49	Renaming File	Excel_Test.xlsx -> E28B7EB1.tmp	₩MSOffice₩E28B7EB1.tmp	Create Attribute	0xB	4
16829192	16:08:49	Renaming File	1B3AC510 -> Excel_Test.xlsx	₩MSOffice₩Excel_Test.xlsx	Create Attribute	0xC	0
16830019	16:08:49	File Deletion		₩MSOffice₩E28B7EB1.tmp	Deallocate File Record Segment	0xB	4
16830789	16:15:51	File Creation		₩MSOffice₩4CA0D510	Initialize File Record Segment	0xB	4
16830989		Writing Content of Non-Resident	Data Runs(in Volume) : 7166(96)	₩MSOffice₩4CA0D510	Update Mapping Pairs	0xB	4
16832436	16:15:52	Renaming File	Excel_Test.xlsx -> 74D8F4FE.tmp	₩MSOffice₩74D8F4FE.tmp	Create Attribute	0xC	0
16832739	16:15:52	Renaming File	4CA0D510 -> Excel_Test.xlsx	₩MSOffice₩Excel_Test.xlsx	Create Attribute	0xB	4
16833472				₩MSOffice₩Excel_Test.xlsx	Update Resident Value	0xB	4
16833544	16:15:52	File Deletion		₩MSOffice₩74D8F4FE.tmp	Deallocate File Record Segment	0xC	0
16833730	16:21:12	File Deletion		₩MSOffice₩~$Excel_Test.xlsx	Deallocate File Record Segment	0xB	6

〈Figure 8〉 Reading and Writing Traces of Excel_Test.xlsx

〈그림 8〉은 예제 파일 F:\MSOffice\Excel_Test.xlsx 문서 열람 및 작성 흔적에 대한 NTFS Log Tracker 1.7 실행 결과이다. 파일 ~$Excel_Test.xlsx의 생성을 나타내는 LSN 16,824,659(Redo: Initialize File Record Segment)과 추가적인 작업없이 해당 파일의 삭제를 나타내는 LSN 16,825,384(Redo: Deallocate File Record Segment)는 파일 Excel_Test.xlsx에 대한 단순 열람 흔적을 나타낸다. 파일 ~$Excel_Test.xlsx의 생성을 나타내는 LSN 16,826,529(Redo: Initialize File Record Segment)는 파일 Excel_Test.xlsx를 다시 오픈하였음을 나타낸다. 이후 1B3AC510 파일 생성을 나타내는 LSN 16,827,274(Redo: Initialize File Record Segment), E28B7EB1.tmp 파일 삭제를 나타내는 LSN 16,830,019(Redo: Deallocate File Record Segment)는 파일 Excel_Test,xlsx에 대한 1차 수정 작업을 종료없이 저장했음을 의미하며, 4CA0D510 파일 생성을 나타내는 LSN 16,830,789(Redo: Initialize File Record Segment) 그리고 파일 74D8F4FE.tmp와 ~$Excel_Test.xlsx 삭제를 나타내는 LSN 16,833,544와 16,833,730(Redo: Deallocate File Record Segment)는 2차 수정 작업후 저장 및 종료하였음을 의미한다.

MS Powerpoint 경우와 마찬가지로 MS Excel에 대한 결과에서도 중간에 생성되는 임시파일들 E28B7EB1.tmp(또는 74D8F4FE.tmp), 1B3AC510(또는 4CA0D510)과 원래의 파일 Excel_Test.xlsx와의 연관성을 보여주는 LSN 16,828,889(또는 16,832,436)과 16,829,192(또는 16,832,739)을 확인할 수 있다.

Ⅵ. MS Office 문서 작성에 따른 파일 크기 변화 조사

Ⅳ장과 Ⅴ장에서 MS Office 문서 파일에 대한 열람 및 작성 흔적을 조사하는 방법에 대하여 기술하였다. 그러나, 앞서 기술한 정도의 작성 흔적만으로는 문서 작성자가 언제 어느 정도 수준으로 해당 문서를 수정하였는지 확인하기 어려운 점이 있다. 게다가 MS Word와 달리 MS Powerpoint와 MS Excel의 경우에는 실제로 작성자가 문서 내용을 수정하지 않고 열람 중 수정없이 단순 저장하는 행위만으로도 파일의 클러스터 할당 정보가 변하고 파일의 실제 크기가 몇백 바이트 정도 변경될 수 있으며 그에 따라 수정시간, 접근시간도 변경된다. MS Word 문서 파일의 경우에도 단지 한 두 글자 정도를 수정하거나 띄어쓰기를 위해 빈칸을 추가하는 정도에도 Ⅳ장과 Ⅴ장과 같은 문서 작성 흔적은 남을 수 있다. 만일 파일에 대한 수정 이후 해당 파일을 저장할 때마다 변화된 파일의 크기를 확인할 수 있다면 보다 정확하게 문서 작성자가 언제 어느 정도 수준으로 해당 문서를 수정하였는지 알 수 있을 것이다. 이번 장에서는 $LogFile에 남아 있는 작업 레코드들을 이용하여 수정되는 파일의 크기가 어떻게 변경되었는지 확인할 수 있는 방법을 기술하고자 한다.

파일의 크기를 확인할 수 있는 $LogFile 작업 레코드들로는 MFT 엔트리 $DATA 속성에서 파일의 클러스터 할당 정보 변경을 기록하는 'Update Mapping Pairs'와 파일의 할당된 크기, 초기화된 크기, 실제 크기 변경을 기록하는 'Set New Attribute Sizes'가 있다. 이러한 작업 레코드들은 파일 생성을 나타내는 작업 레코드 'Initialize File Record Segment' 이후 해당 MFT 엔트리내 관련 정보를 수정하는 과정에서 나

타난다.

Ⅳ장과 Ⅴ장에서 소개한 3개의 예제 파일 Word_Test.docx, PowerPoint_Test.pptx, Excel_Test.xlsx은 각각 2차례에 걸친 수정 작업을 저장하였다. 따라서, 각 파일별로 클러스터 할당 정보가 2차례 변경되고 파일의 실제 크기도 2차례 변경될 것이다. 본 논문에서는 Word_Test.docx 파일을 중심으로 클러스터 할당 정보와 파일의 실제 크기 정보 변경 기록을 조사하는 방법에 대하여 기술하고자 한다.

Ⅴ장의 〈그림 6〉에서 1차 수정 작업을 저장한 파일 ~WRD0000.tmp의 클러스터 할당 정보를 기록하는 LSN 16,802,853(Redo: Update Mapping Pairs)과 16,803,311(Redo: Update Mapping Pairs) 2개가 관찰되고 있으나, 1차 수정 작업 저장이 완료된 후 해당 파일명이 Word_Test.docx로 변경되면 파일에 최종적으로 할당되는 클러스터 정보는 16,803,311에서 얻을 수 있다(LSN 16,802,853에 기록된 클러스터 할당 정보가 갖는 의미에 대해서는 추가적인 연구가 필요하다). 그리고 파일의 실제 크기 정보는 LSN 16,803,413(Redo: Set New Attribute Sizes)에서 얻을 수 있다. 작업 레코드 'Set New Attribute Sizes'는 파일의 할당된 크기 변경, 파일의 초기화된 크기 변경, 파일의 실제 크기 변경을 위하여 3번에 걸쳐 차례로 사용되기 때문에 파일의 실제 크기를 얻기 위해서는 마지막에 기록되는 'Set New Attribute Sizes' 작업 레코드를 조사하여야 한다. LSN 16,803,413의 Redo Data를 조사하면 ~WRD0000.tmp 파일의 할당된 크기 352,256 (0x0000000000056000)바이트(4,096 x 86 = 352,256), 파일의 실제 크기 349,642 (0x00000000000555CA)바이트, 파일의 초기화된 크기 349,642 (0x00000000000555CA)바이트를 구할 수 있다.

다음의 〈표 4〉, 〈표 5〉, 〈표 6〉는 Word_Test.docx에 대한 2차 수정 작업을 저장하는 파일 ~WRD0002.tmp의 클러스터 할당 정보 변경을 기록하고 있는 LSN 16,806,555와 16,807,032, 그리고 파일의 실제 크기 정보 변경을 기록하고 있는 LSN 16,807,134에 대한 실제 데이터 값을 $Logfile 분석 도구 logfileparse.py를[21] 이용하여 추출한 결과이다(표에서 b' '는 십육진수를 리틀 엔디언으로 표기했음을 의미). LSN 16,806,555, 16,807,032, 16,807,134에 적용되는 MFT 엔트리 번호는 모두 4 x Target VCN + Cluster Index / 2 = 4 x 11 + 0 / 2 = 44이며, 이것은 ~WRD002.tmp 파일의 MFT 엔트리로써 2차 파일 수정 작업 저장이 완료되면 Word_Test.docx로 파일명이 변경된다. LSN 16,806,555의 Redo Data로부터 ~WRD0002.tmp 파일 내용이 클러스터 6,864(0x!AD0)에서 시작하여 448개(0x01C0)에 할당되었음을 나타내지만 이것은 ~WRD0002.tmp 파일의 최종적인 클러스터 할당 정보가 아니다. 최종적인 클러스터 할당 정보는 2번째 'Update Mapping Pairs' 로그 레코드를 나타내는 LSN 16,807,032의 Redo Data에서 얻을 수 있으며, 클러스터 6,864(0x1AD0)에서 시작하여 271개(0x010F)가 할당되었음을 확인할 수 있다. 이러한 클러스터 할당 정보는 HDD, USB 메모리와 같은 저장매체에서 수정되기 이전 상태의 파일 내용을 복구하는데 사용될 수 있다.

〈그림 6〉에서 NTFS Log Tracker 1.7은 1차 수정 작업을 저장하는 파일 ~WRD0000.tmp의 경우에는 2개의 'Update Mapping Pairs' 작업 레코드들(LSN 16,802,853 & 16,803,311)에 대한 결과를 보여 주었지만, 2차 수정 작업을 저장하는 파일 ~WRD0002,tmp의 경우에는 1개의 'Update Mapping Pairs' 작업 레코드(LSN 16,806,555)만을 보여 주고 있으나 최종적인 클러스터 할당 정보는 2번째 'Update Mapping Pairs' 작업 레코드 LSN 16,807,032에 기록되어 있다. NTFS Log Tracker 1.7은 대체로 첫 번째 'Update Mapping Pairs' 작업 레코드 결과만을 보여 주기 때문에 NTFS Log Tracker 1.7 결과만으로는 해당 파일의 정확한 클러스터 할당 정보를 얻을 수 없는 경우가 더 많다. 또한, LSN 16,807,134의 Redo Data로부터 직접 ~WRD0002.tmp 파일의 할당된 크기 1,110,016 (0x000000000010F000)바이트(4,096 x 271 = 1,110,016), 파일의 실제 크기 1,107,540 (0x000000000010E654)바이트, 파일의 초기화된 크기 1,107,540 (0x000000000010E654)바이트 정보를 얻을 수 있지만, 〈그림 6〉에서 보듯이 NTFS Log Tracker에서는 이러한 정보를 확인할 수 없다.

파일 Word_Test.docx와 같은 방법으로, 파일 PowerPoint_Test.pptx의 1차 수정 작업 이후 저장된 파일의 클러스터 할당 정보는 LSN 16,814,570 & 16,815,036에서 확인할 수 있으며, 실제 파일의 크기는 LSN 16,815,148에서 확인할 수 있고, 2차 수정 작업이 저장된 파일의 클러스터 할당 정보는 LSN 16,818,082 & 16,818,548에서 확인할 수 있으며, 실제 파일의 크기는 LSN 16,818,638에서 확인할 수 있다. 또한, 파일 Excel_Test.xlsx의 1차 수정 작업 이후 저장된 파일의 클러스터 할당 정보는 LSN 16,827,474 & 16,827,937에서 확인할 수 있으며, 실제 파일의 크기는 LSN 16,828,027에서 확인할 수 있고, 2차 수정 작업이 저장된 파일의 클러스터 할당 정보는 LSN 16,830,989 & 16,831,444에서 확인할 수 있으며, 실제 파일의 크기는 LSN 16,831,543에서 확인할 수 있다.

〈Table 4〉 LSN 16,806,555 (Redo&Undo: Update Mapping Pairs)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
This LSN: 16,806,555(b'9b72000100000000')								Previous LSN: 16,806,538(b'8a72000100000000')							
Client Undo LSN: 16,806,538(b'8a72000100000000')								Client Data Length 56(b'38000000')				Seq. No. 0(b'0000')		Client ID 0(b'0000')	
Record Type Transact. (b'01000000')				Transaction ID 64(b'40000000')				Flags b'0000'		Reserved: b'000000000000'					
Redo Op. b'0900'		Undo Op. b'0900'		RedoOffset 40(b'2800')		Redo Len. 8(b'0800')		UndoOffset 48(b'3000')		Undo Len. 8(b'0800')		TargetAttr 24(b'1800')		LCNtoFoll 1(b'0100')	
RecOffset 272 (b'1001')		AttrOffset 64(b'4000')		MFT Clust. Index 0(b'0000')		Alignment 2(b'0200')		Target VCN 11(b'0b000000')				Alignment b'00000000'			
Target LCN 786,443 (b'0b000c00')				Alignment b'00000000'				Redo Op 0x0009: Update Mapping Pairs Undo Op 0x0009: Update Mapping Pairs							
Redo Data(8 Bytes): 22 C0 01 D0 1A 00 00 00 Undo Data(8 Bytes): 00 00 00 00 00 00 00 00															

〈Table 5〉 LSN 16,807,032 (Redo&Undo: Update Mapping Pairs)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
This LSN: 16,807,032 (b'7874000100000000')								Previous LSN: 16,807,020 (b'6c74000100000000')							
Client Undo LSN: 16,806,538(b'8a72000100000000')								Client Data Length 56(b'38000000')				Seq. No. 0(b'0000')		Client ID 0(b'0000')	
Record Type Transact. (b'01000000')				Transaction ID 64(b'40000000')				Flags b'0000'		Reserved: b'000000000000'					
Redo Op. b'0900'		Undo Op. b'0900'		RedoOffset 40(b'2800')		Redo Len. 7(b'0700')		UndoOffset 48(b'3000')		Undo Len. 7(b'0700')		TargetAttr 24(b'1800')		LCNtoFoll 1(b'0100')	
RecOffset 272 (b'1001')		AttrOffset 65(b'4100')		MFT Clust. Index 0(b'0000')		Alignment 2(b'0200')		Target VCN 11(b'0b000000')				Alignment b'00000000'			
Target LCN 786,443 (b'0b000c00')				Alignment b'00000000'				Redo Op 0x0009: Update Mapping Pairs Undo Op 0x0009: Update Mapping Pairs							
Redo Data(7 Bytes): 0F 01 D0 1A 00 00 00 Undo Data(7 Bytes): C0 01 D0 1A 00 00 00															

〈Table 6〉 LSN 16,807,134 (Redo&Undo: Set New Attrbute Sizes)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
This LSN: 16,807,134 (b'de74000100000000')								Previous LSN: 0 (0x00000000000000)							
Client Undo LSN: 0 (0x00000000000000)								Client Data Length 88(b'58000000')				Seq. No. 0(b'0000')		Client ID 0(b'0000')	
Record Type Transact. (b'01000000')				Transaction ID 64(b'40000000')				Flags b'0000'		Reserved: b'000000000000'					
Redo Op. b'0b00'		Undo Op. b'0b00'		RedoOffset 40(b'2800')		Redo Len. 24(b'1800')		UndoOffset 64(b'4000')		Undo Len. 24(b'1800')		TargetAttr 24(b'1800')		LCNtoFoll 1(b'0100')	
RecOffset 272 (b'1001')		AttrOffset 0(b'0000')		MFT Clust. Index 0(b'0000')		Alignment 2(b'0200')		Target VCN 11(b'0b000000')				Alignment b'00000000'			
Target LCN 786,443 (b'0b000c00')				Alignment b'00000000'				Redo Op 0x000B: Set New Attribute Sizes Undo Op 0x000B: Set New Attribute Sizes							
Redo Data(24 Bytes): 00 F0 10 00 00 00 00 00 54 E6 10 00 00 00 00 00 54 E6 10 00 00 00 00 00 Undo Data(24 Bytes): 00 F0 10 00 00 00 00 00 00 00 00 00 00 00 00 00 54 E6 10 00 00 00 00 00															

Ⅶ. USB 메모리에서 MS Office 문서 열람 및 작성 흔적

USB NTFS 볼륨내에서 MS Powerpoint와 MS Excel 문서 파일을 열람하거나 수정하는 경우는 $LogFile에 기록되는 정보가 SSD, HDD와 동일하지만, MS Word 파일들을 열람하거나 수정하는 경우 해당 볼륨내 $LogFile에 남는 작업 레코드들은 SSD, HDD와 차이가 있다. 일단 USB 저장매체내 NTFS 볼륨에서는 MS Word 파일들을 단순 열람하든, 수정없이 저장하든, 수정한 이후 저장을 하든 어떤한 경우에도, 파일을 오픈할 때 생성되었다가 파일을 종료할 때 삭제되는 ~$*.docx(예를 들어, ~$rd_Test.docx)와 같은 임시파일이 생성되거나 삭제되는 작업 레코드가 존재하지 않는다. 따라서, 저장하는 행위없이 단순 열람만 하는 경우에는 해당 문서를 열람하였는지 확인하기 어렵다. 또한, SSD, HDD의 경우와 달리, MS Word 문서 파일을 열어 어떠한 수정도 하지 않고 단순 저장만 하여도 파일의 클러스터 할당 정보, 파일 크기, 파일 수정/접근 시간이 변경된다. 결과적으로 USB 저장매체내 NTFS 볼륨에서 MS Word 파일을 수정없이 단순 저장하는 경우 파일을 수정하여 저장하는 경우와 동일하게 ~WRDXXXX.tmp, ~WRLXXXX.tmp와('XXXX'는 16진수 4자리를 의미) 같은 임시파일이 생성되었다가 삭제되며 전반적인 작업 진행 과정도 수정하고 저장하는 경우와 동일하게 진행된다. 수정없이 저장을 하든 수정한 이후에 저장을 하든 변경되는 파일 클러스터 할당 정보, 파일 크기, 파일 수정시간 등은 Ⅳ장과 Ⅵ장에서 기술한 방법을 사용하여 얻을 수 있으며, 수정없이 저장하는 경우 변화하는 파일의 크기는 수백 바이트 정도이다.

Ⅷ. 결론

본 논문에서는 NTFS $LogFile을 이용하여 확장자 docx, pptx, xlsx를 갖는 MS Office 문서 파일의 열람 및 작성 흔적을 조사하는 방법을 제시하였다. 볼륨 C:\에 위치한 $LogFile의 경우는 윈도우 운영체제의 시스템 변경을 기록하여야 하기 때문에 3~4시간 정도의 기록밖에는 유지하지 못하지만 볼륨 C:\가 아닌 기타 볼륨에 위치한 $LogFile은 비교적 긴 3일에서 15일 정도의 기록을 유지하기 때문에 C:\ 볼륨이 아닌 볼륨에서 사용자(피의자)가 MS Office 문서 작업을 하였을 경우 포렌식 수사에서 유용하게 활용될 수 있을 것으로 생각된다. 다음은 MS Office 문서 파일 열람 및 작성 작업이 $LogFile에 남기는 흔적과 관련 본 논문의 주요 결과이다:

- docx, pptx, xlsx 문서 파일별로 파일 오픈시 고유한 형태의 임시파일을 생성하고 파일 종료시 해당 임시파일을 삭제하는 흔적을 $LogFile에 남기기 때문에 파일 오픈과 종료 시간을 확인할 수 있다.
- 파일 수정 및 저장시에도 docx, pptx, xlsx 문서 파일별로 각각 고유한 형태의 임시파일(상기 임시파일과는 다른)들을 생성하고 삭제한 흔적을 $LogFile에 남기기 때문에 파일 수정 및 저장 시간을 확인할 수 있다.
- 수정 작업으로 파일 크기가 변경될 경우 $LogFile에 남겨진 작업 레코드를 분석하면 파일의 크기가 수정 과정에서 어떻게 변하였는지 확인 가능하기 때문에 문서 파일 작성에서 사용자가 언제 어느 정도 수준으로 문서를 수정하였는지 확인할 수 있다.
- 상기의 기록들은 단지 가장 최근에 사용한 한번의 작업 기록이 아니라 $LogFile이 해당 기록을 유지하고 있는 기간 동안에 발생한 모든 MS Office 문서 작업 기록을 추적할 수 있도록 해준다.
- MS Powerpoint와 MS Excel 문서 파일은 SSD, HDD, USB 메모리 저장매체 모두 동일한 작업 흔적을 남기는 것으로 파악되었으나 USB 저장매체내 NTFS 볼륨의 MS Word 문서 파일의 경우는 SSD, HDD와는 다른 흔적을 남긴다(본문 Ⅶ장 참조).

확장자 doc, ppt, xls를 갖는 이전 MS Office 문서 포맷을 갖는 문서 파일들도 docx, pptx, xlsx 파일과 유사한 작업 흔적을 남길 것으로 생각되나 추가적인 확인이 필요하다. 또한, 국내에서 많이 사용되고 있는 한컴오피스 HWP 확장자를 갖는 문서 파일의 경우, NTFS Log Tracker 1.7 조사 결과만으로는 파일 오픈이나 종료를 나타내는 어떠한 흔적도 찾을 수 없었으며 파일 수정 및 저장시에는 고유한 형태의 임시파일 $tmXXXX.tmp('XXXX'는 16진수 4자리를 의미)이 생성되었다가 삭제되는 것을 확인할 수 있었으나 보다 상세한 내용은 추가적인 연구가 필요하다.

참 고 문 헌 (References)

[1] Brian Carrier, "File System Forensic Analysis", 2005.0
[2] NTFS.com, "NTFS Transaction Joural", https://www.ntfs.com/transaction.htm
[3] forensic-prooof.com, "LNK (Windows Shortcut) Forensic Analysis," http://forensic-proof.com/archives/607
[4] forensic-prooof.com, "Windows 7 Jump Lists," http://forensic-proof.com/archives/1904
[5] http://www.forensic-artifact.com/windows-forensics/linkfile
[6] http://www.forensic-artifact.com/windows-forensics/jumplist
[7] Supreme Court of Korea, 2015Do2625, 2015.07.16.
[8] Gyu-Sang Cho, "An Analysis of NTFS Journal File for a Computer Forensic", Journal of Digital Forensics, No. 4, pp.40-49, 2009.08. https://www.dbpia.co.kr/
[9] Tae-Han Kim, "Computer Forensic Analysis with Reverse Engineering of NTFS Journal File," The Graduate School of Dongyang University, Ph.D. Thesis, 2010.06.
[10] Tae-Han Kim and Gyu-Sang Cho, "A Digital Forensic Method for File Creation using Journal File of NTFS File System," Journal of the Korea Society of Digital Industry and Information Management Vol.6, No.2, pp.107 - 118, 2010,
[11] Gyu-Sang Cho, "A computer forensic method for detecting timestamp forgery in NTFS", Computers & Security, Volume 34, pp 37-46, 2013.
[12] Dong Bin Oh, Kyung Ho Park and Huy Kang Kim, "De-Wipimization: Detection of data wiping traces for investigating NTFS file system", Computers & Security, Volume 99, pp.1-16, 2020
[13] Junghoon Oh and Hyunuk Hwang, "A Study on the Detection of Suspicious User Behavior through NTFS Log Analysis," Journal of Digital Forensics, Vol. 15, No. 3, pp.54-71, 2021.09.
[14] Junghoon Oh, "NTFS Log Tracker", https://sites.google.com/site/forensicnote/ntfs-log-tracker
[15] Eunjin Kim, Kibom Kim and Hyunuk Hwang, "How To Identify Accessed Files Based On NTFS Filesystem," Journal of Digital Forensics, Vol. 11, No. 2, pp.1-16, 2017
[16] Serim Kang, Soram Kim, Jaehyung Cho and Jongsung Kim, "Development of Integrated Jump Lists Parser and Its Utilization in Forensic," Journal of Digital Forensics, Vol. 11, No. 1, pp.1-15, 2017.06.
[17] KiMin Seo, Hyuk-don Kwon, Jae-wan Bang, Ki-sik Chang and Seung-Jin Oh, "The Analysis of Time Information from Microsoft Office Document in the NTFS File System", Journal of digital forensics, Vol.3, No.2, pp.13-26, 2009.11.
[18] Seyool Park and Sangjin Lee, "For ensic Investigation of HWP File," Journal of Digital Forensics, Vol. 14, No. 4, pp.408-425, 2020.12.
[19] YeonJoo Lee, JeongMin Kim, SungJin Lee, "A Study of Polar is Office Forensic Artifact," Journal of Digital Forensics, Vol. 14, No. 4, pp.368-378, 2020.12.
[20] https://github.com/NTFSparse/ntfs_parse/blob/master/logfileparse.py
[21] https://github.com/jschicht/LogFileParser

저 자 소 개

정 재 훈 (Jaehoon Jeong)
준회원

2022년 2월 서울대학교 수리정보학과 석사
2020년 2월 ~ 2021년 8월 수원고등검찰청 디지털 포렌식팀
2020년 9월 ~ 2022년 2월 수원지방검찰청 안산지청 수사과
2022년 2월 ~ 현재 : 대검찰청 정보통신과
관심분야 : 디지털 포렌식, 정보보호 등

박 상 준 (Sangjoon Park)
정회원

1999년 2월: 성균관대학교 정보통신공학과 박사
1986년 1월 ~ 1999년 12월: 한국전자통신연구원 선임연구원
2000년 1월 ~ 2000년 10월: 국가보안기술연구소 책임연구원
2000년 11월 ~ 2008년 12월: ㈜비씨큐어 부사장
2013년 9월 ~ 현재: 서울대학교 수리정보과학과 객원교수
관심분야 : 디지털 포렌식, 정보보호 등