ISSN(Print): 2287-5891

http://dx.doi.org/10.3745/KTCCS.2013.2.2.093

Record File Carving Technique for Efficient File Recovery in Digital Forensic Investigation

Minsu Park† · Jungheum Park†† · Sangjin Lee†††

ABSTRACT

These days digital data have become essential for digital investigation because most of the crime was occurred by using the digital devices. However, digital data is very easier to falsify or delete. If digital data was deleted, it is necessary to recover the deleted data for obtain digital evidence. Even though file carving is the most important thing to gather. digital evidence in digital forensic investigation, most of popular carving tools don't contemplate methods of selection or restoration for digital forensic investigation. The goal of this research is suggested files which can obtain useful information for digital forensic investigation and proposed new record file carving technique to be able to recover data effectively than before it.

Keywords : Digital Forensics, File Carving, Digital Investigation

디지털 포렌식 조사에서 효율적인 파일 복구를 위한 레코드 파일 카빙 기법

박 민 수† · 박 정 흠†† · 이 상 진†††

요 약

최근 대부분의 범죄에 디지털 매체가 사용되면서 디지털 데이터는 필수 조사 대상이 되었다. 하지만 디지털 데이터는 비교적 쉽게 삭제 및 변조가 가능하다. 따라서 디지털 증거 획득을 위해 삭제된 데이터의 복구가 필요하며, 파일 카빙은 컴퓨터 포렌식 조사에서 증거를 획득할 수 있는 중요한 요소이다. 하지만 현재 사용되는 파일 카빙 도구들은 포렌식 조사를 위한 데이터의 선별을 고려하지 않고 있다. 또 기존의 파일 카빙 기법들은 파일의 일부 영역이 덮어써지거나 조각날 경우 복구가 불가능한 단점이 있다. 따라서 본 논문에서는 포렌식 조사시 유용한 정보를 획득할 수 있는 파일을 제안하고, 기존의 파일 카빙 기법보다 효과적으로 데이터를 복구할 수 있는 레코드 파일 카빙 기법을 제시한다.

키워드 : 디지털 포렌식, 파일 카빙, 디지털 조사

1. 서 론

디지털 기기의 발달과 보급으로 기존의 혈흔, 지문, 범행 도구 등과 같은 아날로그 증거뿐만 아니라 저장된 파일, 시스템 로그 등과 같은 디지털 증거가 중요시 되고 있다. 디지털 증거는 디지털 매체에 파일 형태로 저장되며, 시스템 관리의 목적이나 사용자의 의도에 의해 삭제될 수 있다. 삭제된 데이터는 저장장치에 남겨져 있을 수 있으며, 해당 데이터는 여러 가지 데이터 복구기법을 이용하여 복구할 수 있다. 현재 이용되는 데이터 복구기법은 크게 파일 시스템의 메타 데이터를 이용한 파일 복구 기법과 파일 고유의 정보를 이용한 파일 카빙 기법으로 나눌 수 있다.

현재 대부분의 파일 복구 도구들은 파일 단위로 복구를 수행하며, 멀티미디어, 문서, 그림 파일 등의 사용자 파일을 대상으로 한다. 기존의 파일 복구 연구도 파일 포맷을 분석하여 파일의 Header/Footer 정보를 이용한 파일 단위 카빙 기법을 이용하였다[2]. 하지만 파일 단위 복구 기법은 파일의 일부 영역이 덮어 써지거나 조각날 경우 데이터를 복구할 수 없는 단점이 있으며, 사용자 파일을 대상으로 복구를 수행할 경우 시스템 사용 흔적, 파일의 메타 데이터, 네트워크 사용 흔적 등의 포렌식 조사에 유용한 정보를 획득할 수 없다.

※ 본 논문은 지식경제부 산업융합원천기술개발사업으로 지원된 연구결과임 [10035157, 실시간 분석을 위한 디지털 포렌식 기술 개발].
† 준 회 원 : 고려대학교 정보보호대학원 석사과정
†† 정 회 원 : 고려대학교 정보보호대학원 박사과정
††† 정 회 원 : 고려대학교 정보보호대학원 교수
논문접수 : 2012년 9월 20일
수 정 일 : 1차 2012년 11월 8일, 2차 2012년 11월 30일
심사완료 : 2012년 11월 30일
* Corresponding Author : Sangjin Lee(sangjin@korea.ac.kr)

따라서 포렌식 조사에 유용한 정보를 획득하기 위해서는 네트워크 패킷, 파일의 메타데이터, 시스템 로그 등 시스템 파일을 포함하여 복구를 수행해야 하며, 이를 위해 레코드 단위 파일 복구를 추가하면 기존의 파일 단위 카빙 기법으로 획득하지 못한 많은 데이터를 복구할 수 있다.

2. 파일 복구 기법

일반적으로 파일은 하드디스크, USB 메모리 등과 같은 디지털 저장매체에 저장된다. 저장된 파일이 삭제 될 때에는 파일의 메타데이터에 삭제 여부를 표시하고, 데이터는 저장매체에 남아있다. 따라서 삭제된 파일은 해당 파일 시스템의 메타데이터를 이용하거나 파일의 고유 정보를 이용하여 복구할 수 있다[2]. 파일 복구 방법은 크게 파일시스템의 메타데이터를 이용한 복구와 다양한 파일들의 고유정보를 이용한 파일 카빙 복구로 나눌 수 있다.

2.1 파일 시스템 정보를 이용한 파일 복구 기법

메타데이터를 이용한 파일 복구는 파일시스템의 특성을 이용한다. 파일시스템은 여러 종류가 존재하며, 일반적으로 Fig. 1과 같은 구조로 파일을 관리한다[1].

Fig. 1. File System

파일 목록에는 파일의 이름, 크기, 위치, 시간 정보 등이 저장되어 있다. 파일을 삭제하면 대부분의 파일 시스템은 파일 내용을 삭제하지 않고, 파일 목록에 해당 파일의 삭제 여부를 표시한다. 따라서 파일 목록에 저장된 정보를 이용하면 삭제된 파일의 내용을 확인할 수 있다. 윈도우에서 널리 사용되는 FAT, NTFS 등의 파일시스템은 모두 이러한 방식을 사용하고 있다. 하지만 파일 목록이 손상된다면 더 이상 파일의 위치를 찾을 수가 없게 되고, 파일의 복구도 불가능하다[2].

2.2 파일 카빙 기법

파일 카빙은 파일 시스템 정보를 사용하지 않고, 파일의 고유 특성만을 이용한 파일 복구 기법을 말하며, 데이터가 덮어 쓰이지 않은 비할당 영역을 대상으로 수행한다. 파일 카빙 기법은 크게 4가지로 나눌 수 있으며, Table 1과 같다[4].

Header/Footer를 이용한 파일 카빙 기법은 특정 파일의 시작 위치를 파악할 수 있는 시그니처 또는 특정 문자열과 같은 정보를 담고 있는 파일 헤더를 탐색하여 파일의 시작 위치를 검색하고, 파일의 끝에 나타나는 파일 고유 정보인 Footer를 검색한 후 파일 Header와 Footer 사이의 모든 데이터를 하나의 특정 파일로 판단하는 복구 기법을 말한다[4].

Table 1. File Carving

카빙 기법 종류	특징
Header/Footer	파일의 Header와 Footer를 검색
Header/ File Size	파일의 Header를 검색하고, Header를 기준으로 특정 위치에 존재하는 File Size나 블록의 수를 계산
Header/ Ram Slack	파일의 Header와 Ram Slack을 검색
파일 구조 검증	파일의 고유 특징 (ASCII, MIME)등을 이용

Header/File Size를 이용한 파일 카빙 기법은 자신의 사이즈를 파일 내부에 저장하고, 있는 파일을 복구 할 수 있는 기법으로 Header 정보를 검색하여 시작위치를 검색한 후 특정 위치에 존재하는 파일 사이즈 정보를 읽거나 파일을 구성하는 블록의 수를 계산하여 파일을 복구하는 방법을 말한다[4].

File Size나 Footer와 같이 파일 고유의 정보를 이용하여 파일의 끝을 알 수 없을 때는 파일 시스템의 특성인 Ram Slack을 이용하여 파일을 복구할 수 있다[4].

파일의 시작을 알 수 있는 Header나 시그니처 정보가 없을 경우 파일의 고유 특징을 이용하여 파일을 검증할 수 있다. 예를 들어 HTML은 <title>, <html>과 같이 <>와 ASCII 코드로 구성되어 있고, EML은 MIME 형식으로 구성되어 있다. 이러한 파일 고유의 특징을 이용하여 파일을 검증 및 복구할 수 있다[4].

2.3 기존의 파일 카빙 기법의 특징 및 한계점

기존의 파일 카빙 기법은 문서 파일, 그림 파일, 멀티미디어 파일 등 사용자 파일을 대상으로 하며, 삭제된 파일을 복구하는데 주로 사용된다. 하지만 포렌식 조사에 유용한 파일의 시간 정보, 시스템에서 발생한 이벤트, 시스템의 ON/OFF 시간, 인터넷 검색 기록 등의 정보는 획득할 수 없다. 따라서 사용자의 행동을 분석하는데 한계가 있다. 이러한 한계를 극복하기 위해서 포렌식 조사에 유용한 개선된 카빙 기법이 필요하다. 특히 포렌식 조사에 유용한 파일인 웹 서비스 사용 흔적, 메모리 내용, 시스템 로그 등의 파일을 복구하면 조사 대상이 되는 기간 동안 사용자가 실행한 작업 내용, 웹 서비스 검색 기록, 시스템 ON/OFF 시간, 설치 또는 삭제된 프로그램 및 시스템 변경 내역 등의 정보를 획득할 수 있다.

3. 포렌식 조사를 위한 레코드 단위 파일 복구 기법

3.1 관련 연구

이와 관련된 연구로는 임성수의 "SQLite 데이터베이스 파일 카빙 방안 연구"와 Robert Beverly의 "Forensic carving of network packets and associated data

structures"가 있다. 임성수는 SQLite에서 실제 데이터가 저장된 Leaf Node를 획득하는 방법을 제시하였고, Robert Beverly는 네트워크 정보가 저장된 IP Packet을 획득하는 방법을 제시하였다[3][8]. 하지만 임성수의 연구는 SQLite 데이터베이스 파일 복구에만 국한되어 있고, Robert Beverly는 Network Packet만을 대상으로 한다. 또한 Rovert Beverly의 연구에서 획득한 데이터는 바이너리 또는 텍스트로 구성된 정보만을 담고 있어 기존의 분석 도구로는 분석할 수 없는 한계가 있다. 따라서 본 논문에서는 효과적인 파일 복구와 복구된 파일의 효율적인 분석을 위해 레코드 단위로 저장하는 다양한 파일들을 조사 및 분석하고, 레코드 단위 파일 카빙 기법을 제시한다.

3.2 포렌식 조사에 유용한 파일

본 논문에서 제안하는 포렌식 조사에 유용한 파일은 Table 2와 같다.

Table 2. Useful in the forensic investigation file (*Record File Carving)

분류	대상 파일	획득 가능한 정보
이메일	EML, PST, DBX	이메일 사용 내역 및 내용
가상 시스템	VMDK, VDI	가상 머신 사용 흔적 및 가상 머신에 저장된 파일
텍스트 파일	SetUpApi	시스템에 설치된 하드웨어 로그
	HTML	방문한 웹 페이지 흔적
실행 파일	EXE	시스템에 저장된 실행 파일
분류	대상 파일	획득 가능한 정보
윈도우 아티팩트	Registry Hive	시스템 정보
	LNK, Prefetch	설치 및 실행된 프로그램
	*Event Log	시스템에서 발생했던 이벤트
	*Change log	설치 또는 삭제된 프로그램 및 시스템 변경 내역
데이터 베이스	EDB	문서, E-Mail, 비디오 등과 같은 시스템에서 사용된 데이터 정보
	*SQLite	스마트폰 데이터 및 인터넷 사용 흔적
파일 시스템	*Directory Entry	FAT 파일 시스템에 저장된 파일 이름 및 디스크 저장 위치
	*MFT Entry	NTFS 파일 시스템에 저장된 파일 이름 및 디스크 저장 위치
	*MFT Entry (Resident)	MFT Entry에 저장된 파일
네트워크	*Packet	네트워크 통신 정보
웹 서비스	*Index.dat	검색어, 방문한 웹 페이지, 방문 시간과 같은 사용자의 인터넷 사용흔적

해당 파일들은 기존의 파일 카빙 기법을 이용하여 복구할 수 있다. 예를 들어 텍스트 파일은 파일 구조 검증을 통해 복구가 가능하고, 이메일과 윈도우 아티팩트, 웹 서비스 파일은 Header/File Size를 이용하여 복구할 수 있다. 하지만 해당 파일이 손상되어 기존의 카빙 기법을 이용하여 복구가 불가능할 경우에는 Event Log와 Change Log, Index.dat 등의 파일은 레코드 단위로 파일 내용이 저장되므로 본 논문에서 제안하는 레코드 단위 파일 카빙 기법을 이용하면 기존의 카빙 기법에서 복구할 수 없었던 파일이 조각나거나 일부 영역이 덮어 쓰인 파일도 복구가 가능하다.

3.3 레코드 단위 파일 카빙 기법

본 논문에서 제안하는 레코드 단위 파일 카빙 기법은 파일을 구성하는 레코드 단위로 데이터를 복구하여 파일의 일부분을 획득하는 방법이다. 레코드란 특정 파일 포맷에서 파일 내용이 저장되는 단위 또는 블록을 의미한다.

1) Index.dat

웹 서비스 사용 흔적이 기록된 index.dat은 Header, Hash Table, Activity Record로 구성되어 있으며, 일부 정보가 덮어써져 파일 카빙 기법으로 복구가 불가능할 때 실제 의미 있는 데이터가 저장되어 있는 Activity Record를 복구한 후 재조합 과정을 거치면 유용한 데이터를 획득할 수 있다.

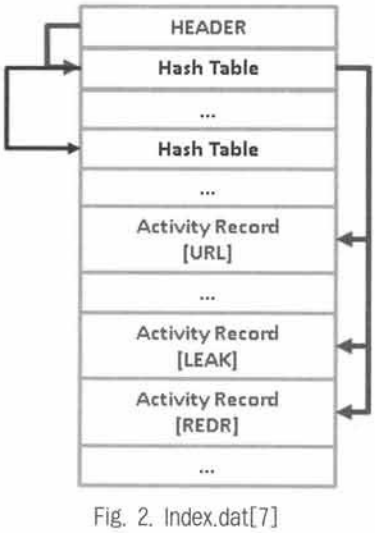

Fig. 2. Index.dat[7]

Index.dat 파일의 구조는 Fig. 2와 같다. Header는 File Size, Version, Hash Table의 시작 위치 등의 정보를 가지고 있으며 Hash Table은 Signature, Block Count, Next Table Offset 정보를 가지고 있다. 만약 Index.dat 파일에서 Header 정보가 유실되었을 경우 남아있는 Activity Record로 구성된 Index.dat 파일을 만들 수 있다. Activity Record는 Signature, Activity Record 크기 (블록의 수)를 가지고 있어 해당 Activity Record를 추출할 수 있다. 추출된 여러 개의 Activity Record를 하나의 파일로 조합하고, Header의

Version, File Size 및 Hash Table의 Signatre, Block Count, Next Table Offset 정보를 결합된 Activity Record에 맞게 구성하여 헤더를 생성할 수 있다. 이렇게 복구된 파일은 Index.dat 파일 포맷과 동일하여 여러 도구를 이용하여 분석할 수 있다.

2) Evt,Evtx

Evt와 Evtx는 윈도우 시스템에서 발생된 이벤트를 저장한 로그파일이다. Evt는 Windows 2000과 2003, XP에서 사용되며, Evtx는 Windows Vista와 7에서 사용된다.

a) EVT

Evt의 구조는 Fig. 3과 같다.

Fig. 3. Evt 구조

Evt는 Header와 Record, Cursor(EOF)로 구성되어 있다. Header에는 Evt 시그니처, FileSize가 저장되어 있고, 해당 정보를 이용하여 Evt 파일을 복구할 수 있다. 만약 Evt가 조각나거나 Header가 덮어써져 기존의 파일 단위 카빙 기법으로 복구할 수 없을 때에는 이벤트 정보가 저장된 Record를 추출하고, Evt Header 및 Cursor를 재구성하면 정상적인 Evt 파일로 복구할 수 있다.

Evt Record의 처음 4Bytes에는 Record의 크기가 저장되어 있고, 다음 4Bytes에는 Record를 구별할 수 있는 'LfLe (0x4C,0x66,0x4C,0x65)'라는 문자열이 저장되어 있다. 따라서 문자열 'LfLe'를 검색하고, Evt Record에 저장된 Data Offset 값과 Data Size 값이 Record 전체 크기 범위내의 값인지 확인한 후 처음 4Bytes에 저장된 Record의 크기만큼 추출하면 Evt 데이터를 획득할 수 있다.

b) EVTX

Evtx의 구조는 Fig. 4와 같다.

Evtx는 Header와 다수의 Chunk 구성되고, 각각의 Chunk는 다수의 레코드로 이루어져 있다. Evtx는 Record에 데이터를 저장하고 있다. 따라서 Evtx의 헤더정보가 손실되어 기존의 파일 단위 카빙 기법으로 복구할 수 없다면 Evtx 레코드를 복구하여 저장된 데이터를 획득하고, 복구한 레코드를 하나의 Chunk로 구성하고, Chunk Header와 Evtx

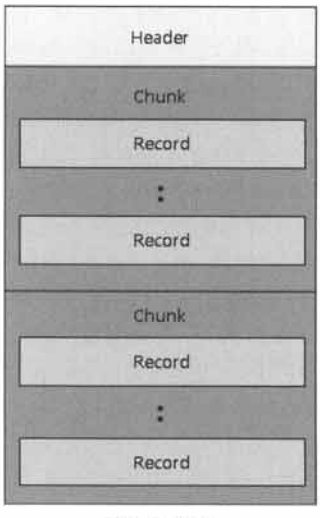

Fig. 4. Evtx

Header를 생성하여 정상적인 Evtx 파일로 복구할 수 있다.

Evtx 레코드는 0x2a,0x2a,0x00,0x00의 고정된 값으로 시작되며, 다음 4바이트에 저장된 레코드의 크기 값을 확인하여 추출할 수 있다. 추출된 레코드를 하나의 Chunk로 조합하고 조합된 Chunk를 재구성하여 하나의 Evtx 파일로 복구할 수 있다. 이때 Evtx File Header의 Magic, Header Size, Chunk Count와 Chunk Header의 Magic, Next Chunk Offset과 같은 필수적인 필드 값을 추출된 레코드의 개수에 맞게 구성해야 한다.

c) MFT Entry

MFT Entry는 시스템에 존재하는 각 파일의 정보를 담고 있는 Entry 하나하나가 레코드라고 할 수 있다. MFT Entry를 복구하기 위해서 Entry 시작 값인 'FILE (0x46,0x49,0x4c,0x45)'을 찾는다. 해당 문자열을 찾았다면 Offset Fixup Array의 값을 확인하여 Fixup Array의 시작 위치를 획득한다. 만약 해당 위치에 저장된 Fixup Array 값이 MFT Entry의 마지막 값과 동일하다면 정상적인 MFT Entry라고 판단한 후 복구할 수 있다.

이때 MFT Entry가 Resident 속성을 가지고 있을 경우 해당 파일이 MFT Entry내에 저장되어 있어 파일 내용을 획득할 수 있다.

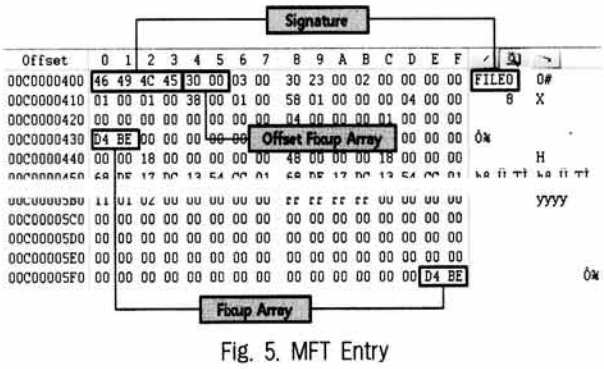

Fig. 5. MFT Entry

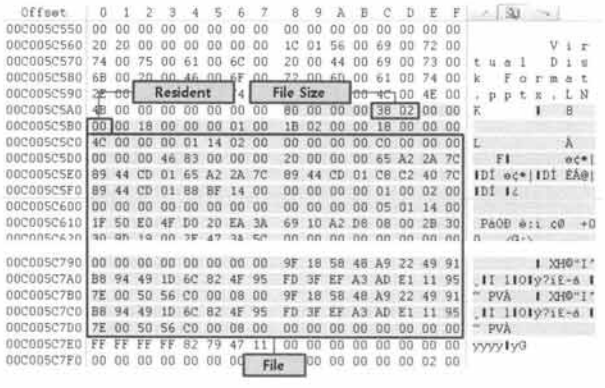

Fig. 6. Acquired files which are stored in the MFT Entry

d) Directory Entry

Directory Entry는 FAT File System에서 사용되고, 디스크에 저장된 디렉터리 및 파일의 이름과 확장자, 속성, 생성 날짜 및 시간, 수정 날짜 및 시간, 실행 날짜, 파일의 크기 정보를 담고 있다. Directory Entry는 32Bytes의 고정된 크기를 가진다. 만약 파일 이름이 8Bytes를 초과할 경우에는 LFN(Long File Name) Entry라는 새로운 구조를 사용하여 파일의 정보를 저장한다. Directory Entry와 LFN Entry의 구조는 Fig. 7과 같다[9].

Directory Entry는 특정한 시그니처 정보를 가지고 있지 않기 때문에 Directory Entry를 구성하는 각 항목의 값이 올바른지 검증해야 한다. 또 모든 디렉토리는 자기 자신(\.)과 상위 디렉토리(\..)를 포함하고 있고, 각 디렉토리의 처음에 위치한다. 따라서 해당 Entry를 검색하고 다음에 위치하는 Entry들의 구조 검증을 통해 복구할 수 있다. 이때 LFN의 경우에는 마지막 Entry의 Name 값에 0xFF가 저장되어 있기 때문에 해당 값을 확인하고, 해당 Entry까지 하나의 LFN으로 추출한다. LFN이 아니라면 32Bytes의 고정된 크기를 가지므로 해당 크기만큼 추출한다.

e) Directory Entry

네트워크 패킷은 파일이 아닌 응용프로그램에서 사용하는 데이터이며, 메모리에 저장된다. 윈도우는 가상 메모리를 관리하기 위해 고정된 위치에 Pagefile을 생성하고, 메모리 내용을 주기적으로 Pagefile에 덮어쓴다. 따라서 Pagefile에는 주고받은 네트워크 패킷이 저장될 수 있으며, 해당 파일을 조사하면 네트워크 패킷을 획득할 수 있다. VM Ware와 Virtual Box와 같은 가상 시스템의 경우에도 Guest OS의 메모리 정보를 파일 형태로 저장하기 때문에 해당 파일을 조사하면 네트워크 패킷을 획득할 수 있다. 가상 시스템의 메모리 파일은 Guest OS를 부팅할 때마다 생성되고, Guest OS를 종료할 때 삭제된다. 따라서 가상 시스템이 사용된 디스크의 경우 비할당 영역에도 네트워크 패킷이 존재할 수 있다.

네트워크 패킷의 구조는 Fig. 8과 같다. 네트워크 패킷을 복구하기 위해서는 각 프로토콜의 Header 구조를 검증해야 하며, 데이터 링크 계층이 덮어 써졌을 경우에는 네트워크 계층을 복구하여 주고받은 Data를 획득할 수 있다[8]. IP 패킷은 네트워크 계층 프로토콜이며, IP Header에는 Version 정보, Header Size와 Total Length, Flag, TTL (Time To Live), Protocol Type 등의 정보가 저장되어 있다. IP 패킷을 복구하기 위해서는 IP Header의 각 필드에 저장된 값의 범위가 올바른지 확인하고, Checksum 필드의 값과 비교한

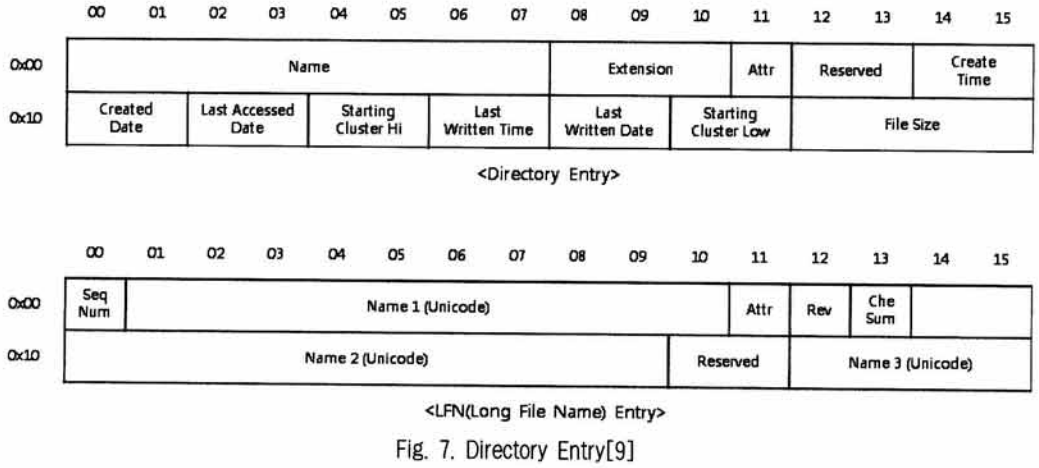

Fig. 7. Directory Entry[9]

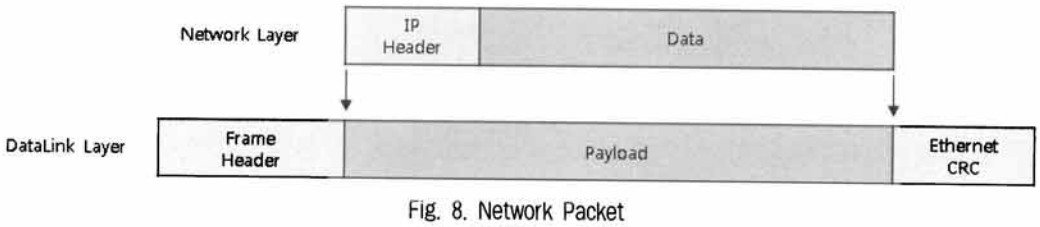

Fig. 8. Network Packet

후 Total Length에 저장된 크기만큼 추출한다. 추출된 IP 패킷은 Wireshark, Tcpdump 등의 도구를 이용하여 효율적으로 분석하기 위해 다수의 IP 패킷들을 네트워크 패킷 캡처 파일로 사용되는 PCAP 파일로 재구성하여 저장한다.

f) SQLite

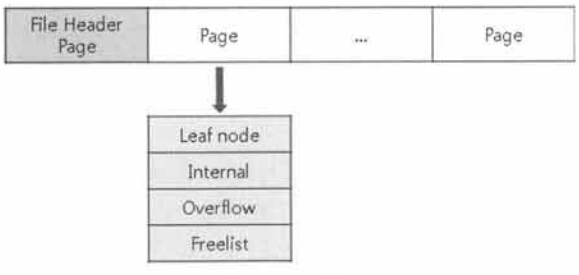

Fig. 9. SQLite

SQLite 파일의 전체 구조는 Fig. 9와 같다. SQLite는 Page라는 저장 단위를 사용하여 데이터를 저장한다. 파일의 처음에 나타나는 Page를 Header Page라고 하며, 파일을 식별할 수 있는 Header String, Page Size 등의 정보가 저장되어 있다. Header Page 이후에 존재하는 Page는 Leaf node, Internal, Overflow, FreeList 이렇게 4가지 종류가 있다. 만약 SQLite 파일이 조각나거나 Header Page가 덮어써져 SQLite 파일의 복구가 불가능할 때에는 실제 데이터베이스 레코드를 저장하고 있는 Leaf node를 복구하여 데이터를 획득할 수 있다. Leaf node의 구조는 Fig. 10과 같다[3].

Leaf node는 페이지 헤더, 레코드 위치 배열, 비할당 영역, 레코드 데이터 영역으로 구성되어 있다. Leaf node는 레코드 단위로 데이터를 관리하며, 각 레코드에는 테이블의 Row 데이터가 저장되어 있다. Leaf node를 복구하기 위해서 페이지 헤더의 Page flag 값을 Leaf node의 시그니쳐로 사용한다. 레코드 위치 배열에는 레코드 데이터 영역에 저장된 각 레코드들의 위치가 저장되어 있다. 레코드 데이터 영역에 저장된 레코드는 레코드 헤더와 레코드 데이터로 구성되고, 레코드 헤더의 크기 값을 확인하여 하나의 레코드 크기를 알 수 있다. 이와 같은 과정을 반복하여 Leaf node에 저장된 전체 레코드의 크기를 계산하고, 해당 데이터를 복구한다[3].

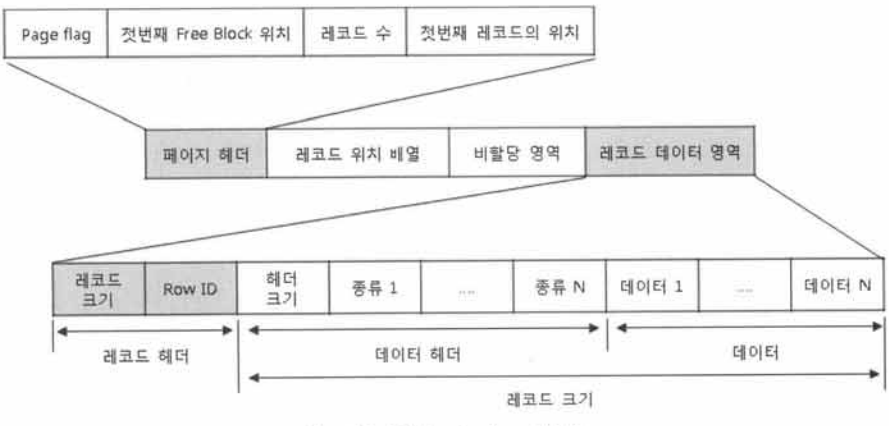

Fig. 10. SQLite Leaf node[3]

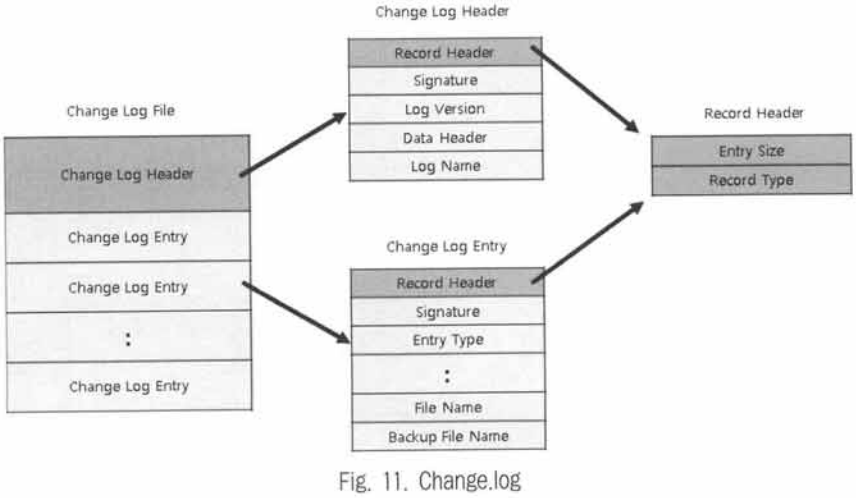

Fig. 11. Change.log

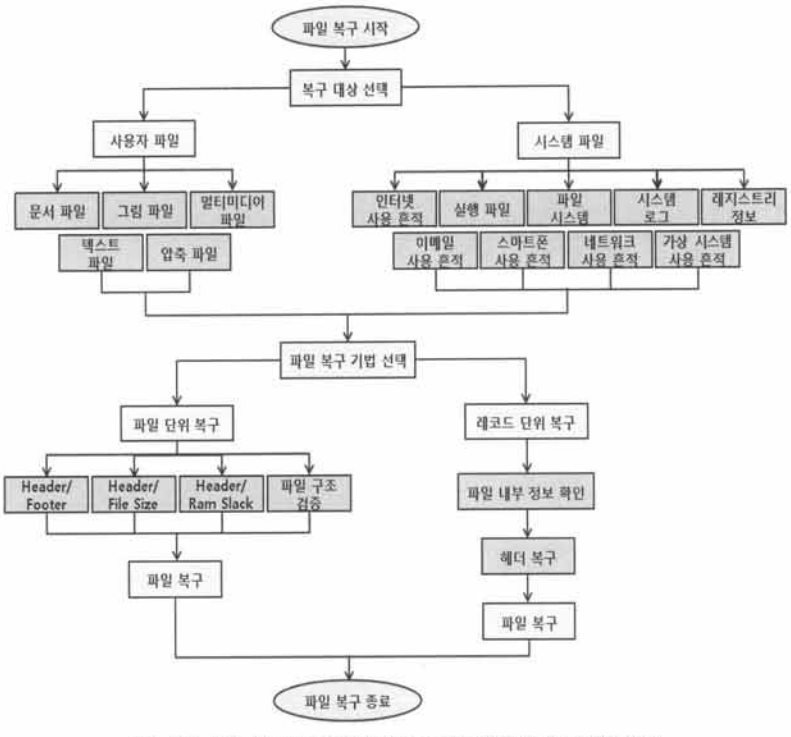

Fig. 12. File Carving Techniques for Digital Investigation

g) Change log

Change log는 시스템 복원에 사용되고, 백업된 파일의 전체 경로, 원본 파일명, 백업 파일명 등이 저장되어 있어 시스템에 저장된 파일 목록을 획득할 수 있다.

Change log는 Fig. 11과 같이 Change Log Header와 Change Log Entry로 구성되어 있고, Change Log Header와 Entry는 동일한 구조의 Record Header로 시작한다. Change Log의 데이터는 Change Log Eantry에 저장되어 있어 해당 Entry를 복구하면 유용한 정보를 획득할 수 있다. Change Log Entry를 복구하기 위해서는 해당 Entry에 저장된 4Bytes의 고정된 Signature(0xABCDEF12) 값과 Entry Type, Entry Flags, Attributes 값의 범위가 올바른지 확인하고, Record Header에 저장된 Entry Size를 계산하여 추출한다.

4. 구현 결과

본 논문에서 제안한 포렌식 조사를 위한 카빙 기법의 전체 구성은 Fig. 12와 같다. 본 논문에서 제안한 레코드 단위 파일 카빙 기법을 개인 PC와 공용 PC에 실험을 하였다. 실험 대상 시스템은 Table 3과 같다. 해당 시스템의 삭제된 데이터를 복구하여 파일 시스템, 웹 서비스 사용기록, 위부 저장장치 사용 흔적, 시스템 ON/OFF 기록, 설치된 프로그램, 최근 사용한 문서 정보 등 사용자 파일에서는 획득할 수 없었던 정보들을 획득할 수 있었다. Table 4는 파일 단위 카빙 기법과 레코드 단위 카빙 기법을 이용하여 복구한 레코드의 수이다. 파일 단위 파일 카빙 기법은 상용 파일 복구 도구인 Recover My Files 5.1.0.1824를 사용하였다[10].

Table 3. Recovery target system

PC 구분	운영체제	파일 시스템	사용 기간	저장장치 크기	비할당 영역 크기	소요시간
개인 PC_1	Windows 7 Enterprise k	NTFS	1년 2개월	350 GB	124 GB	2시간 10분
개인 PC_2	Windows 7 Enterprise k	NTFS	2년 6개월	500 GB	72 GB	1시간 50분
개인 PC_3	Windows 2003	NTFS	4개월	500 GB	102 GB	1시간 58분
개인 PC_4	Windows XP Professional	NTFS	9개월	500 GB	390 GB	5시간 12분
개인 PC_5	Windows XP Professional	FAT32	1개월	120 GB	52 GB	1시간 12분
공용 PC_1	Windows 7 Home edition	NTFS	7개월	500 GB	359 GB	5시간 55분
공용 PC_2	Windows 7 Home edition	NTFS	1년 6개월	500 GB	270 GB	4시간 12분
공용 PC_3	Windows XP Professional	NTFS	2년 4개월	350 GB	120 GB	2시간 8분
공용 PC_4	Windows XP Professional	NTFS	2년 4개월	350 GB	156 GB	2시간 36분

Table 4. Record file carving

종류	대상 시스템					
	개인 PC1(Windows 7)		개인 PC2(Windwos 7)		개인 PC3(Windows 2003)	
	파일 단위 카빙 기법	레코드 단위 카빙 기법	파일 단위 카빙 기법	레코드 단위 카빙 기법	파일 단위 카빙 기법	레코드 단위 카빙 기법
Index.dat (History)	837	6472	592	3812	465	4238
EVTX (Event)	N/A	72	N/A	80	N/A	0
EVT (Event)	N/A	0	N/A	0	N/A	0
Change Log	N/A	0	N/A	0	N/A	0
MFT Entry	N/A	78562	N/A	96547	N/A	10873
Directory Entry	N/A	0	N/A	0	N/A	0
Network Packet	N/A	270	N/A	412	N/A	26
SQLite (Record)	N/A	9	N/A	0	N/A	7

종류	대상 시스템					
	개인 PC4(Windows XP)		개인 PC5(Windows XP)		공용 PC1(Windows 7)	
	파일 단위 카빙 기법	레코드 단위 카빙 기법	파일 단위 카빙 기법	레코드 단위 카빙 기법	파일 단위 카빙 기법	레코드 단위 카빙 기법
Index.dat (History)	766	5726	45	45	560	3062
EVTX (Event)	N/A	0	N/A	0	N/A	41
EVT (Event)	N/A	21	N/A	7	N/A	15
Change Log	N/A	8	N/A	5	N/A	12
MFT Entry	N/A	0	N/A	0	N/A	58454
Directory Entry	N/A	146	N/A	132	N/A	0
Network Packet	N/A	45	N/A	72	N/A	130
SQLite (Record)	N/A	0	N/A	0	N/A	0

종류	대상 시스템					
	공용 PC2(Windows 7)		공용 PC3(Windows XP)		공용 PC4(Windows XP)	
	파일 단위 카빙 기법	레코드 단위 카빙 기법	파일 단위 카빙 기법	레코드 단위 카빙 기법	파일 단위 카빙 기법	레코드 단위 카빙 기법
Index.dat (History)	812	6521	1077	7811	994	5605
EVTX (Event)	N/A	32	N/A	0	N/A	0
EVT (Event)	N/A	0	N/A	19	N/A	8
Change Log	N/A	0	N/A	2	N/A	0
MFT Entry	N/A	74572	N/A	0	N/A	0
Directory Entry	N/A	0	N/A	71	N/A	59
Network Packet	N/A	34	N/A	23	N/A	26
SQLite (Record)	N/A	0	N/A	0	N/A	0

비교 대상인 Recorver My Files는 Table 4의 레코드 단위 파일 카빙 기법을 적용할 수 있는 파일들 중 Index.dat 파일 복구만을 지원하고, MFT Entry, Change Log, MFT Entry와 같은 다른 시스템 파일의 복구는 지원하지 않는다. 따라서 Table 4의 파일 단위 카빙 기법을 이용하여 복구 작업 시 Index.dat 파일 복구 결과만 존재한다. 하지만 본 논

문에서 제안한 레코드 단위 파일 카빙 기법을 이용하면 해당 시스템 파일들을 효율적으로 복구할 수 있다.

레코드 단위 파일 카빙 기법을 통해 파일 복구 작업을 수행한 결과 개인 PC1은 파일 단위 카빙 기법보다 약 8배 많은 6472개의 Index.dat 레코드를 복구할 수 있었다. EVTX는 약 72개의 이벤트를 복구하였고, MFT Entry는 78562개의 Entry를 복구할 수 있었다. Network Packet은 270개를 복구하였고, SQLite는 9개를 복구하였다. 파일 복구 결과를 확인하면 Index.dat 파일과 MFT Entry가 가장 많이 복구되는 것을 확인할 수 있다. 해당 파일들은 파일의 생성과 삭제, 인터넷 방문 및 검색 기록 등과 같이 사용자의 행동 및 소유한 데이터를 분석하는데 매우 유용한 파일이다. 개인 PC2의 경우 파일 단위 카빙 기법보다 약 6배 많은 3812개의 Index.dat 파일을 복구할 수 있었다. EVTX의 경우 80개의 이벤트를 복구할 수 있었고, MFT Entry는 96547개를 복구할 수 있었다. Network Packet은 412개를 복구하였다. 개인 PC4와 개인 PC5는 앞에서 설명한 개인 PC1과 개인 PC2와는 다르게 FAT 파일 시스템을 사용하기 때문에 MFT Entry가 아닌 146개의 Directory Entry가 복구되었다. 또 Windows XP를 사용하여 EVTX가 아닌 EVT가 복구되었고, 각각 21개와 7개의 이벤트가 복구 되었다. 공용 PC1의 경우에는 Windows 7을 사용하고 있어 41개의 EVTX를 복구할 수 있었다. 하지만 XP에서 사용하는 EVT도 15개 복구할 수 있었다. 해당 PC의 복구된 레지스트리 파일을 분석한 결과 해당 시스템은 현재 Windows 7이 설치되어 사용되고 있지만 이전에 Windows XP가 설치된 것을 확인할 수 있었다. 나머지 PC의 파일 복구 결과는 Table 4와 같다. 이와 같이 레코드 단위 파일 카빙 기법을 이용하면 다양한 시스템 파일을 효율적으로 복구할 수 있다. 또한 레코드 카빙 기법을 통해 복구된 Index.dat와 EVTX, 네트워크 패킷 등의 레코드들을 재조합하고 헤더를 재구성하여 하나의 파일로 복구하였고, 하나의 파일로 복구된 파일들은 WEFA, Tcpdump, Wireshark, 등의 도구를 이용하여 분석할 수 있었다[11][12][13].

이와 같이 본 논문에서 제안한 포렌식 조사에 유용한 파일을 복구하면 사용자 파일에서는 획득할 수 없었던 유용한 정보들을 획득할 수 있고, 레코드 단위 카빙 기법을 통해 기존의 파일 단위 카빙 기법으로는 복구할 수 없었던 데이터를 복구할 수 있다.

5. 결 론

본 논문은 포렌식 조사를 위한 데이터 복구 기법으로 2가지 방안을 제시하였다. 첫째, 포렌식 조사에 유용한 파일을 선별하였다. 둘째, 파일의 내용이 저장된 레코드를 복구하는 레코드 단위 파일 카빙 기법을 제시하였다. 구현 결과 사용자 파일에서 획득할 수 없었던 사용자의 행위 또는 시스템 사용 기록 등을 획득할 수 있었고, 기존의 카빙 도구에서 복구할 수 없었던 일부 영역이 덮어 써지거나 조각난 파일의 데이터도 획득할 수 있었다. 게다가, 레코드 단위 파일 카빙 기법을 통해 복구된 데이터의 헤더를 재구성하여 기존의 분석 도구들과 호환성도 유지하였다.

앞으로 더 많은 정보의 획득을 위해서는 새로운 파일에 대한 발견 및 연구가 필요하다. 또한 레코드로 구성되지 않은 파일에 대한 개선된 카빙 기법을 연구하여 기존의 카빙 기법에서 복구할 수 없는 데이터를 획득할 수 있는 방법이 필요하다.

참 고 문 헌

[1] Brian Carrier, "File System Forensic Analysis", Addison Wesley Professional, 2005.

[2] Jinkook Kim, "A Framework for Data Recovery and Analysis from Digital Forensics Point of View", Korea Information Processing Society. Vol.17-C, No.5, pp.391-398, 2010.

[3] Sungsu Lim, "The Research on File Carving Method of SQLite Database", 2010 The Workshop of Digital Forensics. pp.79-82, 2010.

[4] Digambar Povar, V.K. Bhadran, "Forensic Data Carving", Digital Forensics and Cyber Crime, Vol.53, pp.137-148, 2011.

[5] M.I Cohen, "Advanced carving techniques", Digital Investigation, Vol.4, Issues.3-4, pp.119-128, 2007.

[7] Junghoon Oh, "A Study for recovering Deleted Information of Web Browser", 2010 The Workshop of Digital Forensics. pp.79-82, 2010.

[8] Robert Beverly, "Forensic carving of network packets and associated data structures", Digital Investigation, Vol.8, pp.78-89, 2011.

[9] Sangjin Lee, "Introduction to Digital Forensics", Eroon. pp.180-181, 2010.

[10] http://www.getdata.com/, GetData

[11] Junghoon Oh "Advanced evidence collection and analysis of web browser activity", Digital Investigation, Vol.8, pp.62-70, 2011.

[12] http://www.tcpdump.org/, TCPDUMP&LiBPCAP

[13] http://www.wireshark.org/, WIRESHARK

박 민 수

e-mail : minsoon2@korea.ac.kr

2010년 신라대학교 컴퓨터정보공학부 (학사)

2010년~현 재 고려대학교 정보보호학과 석사과정

관심분야 : 디지털 포렌식, 역공학, 보안성평가

박 정 흠

e-mail : junghmi@korea.ac.kr
2007년 한양대학교 컴퓨터전공(학사)
2009년 고려대학교 정보보호학과(석사)
2009년~현 재 고려대학교 정보보호대학원 박사과정
관심분야 : 디지털 포렌식, 안티-안티 포렌식

이 상 진

e-mail : sangjin@korea.ac.kr
1987년 고려대학교 수학과(학사)
1989년 고려대학교 수학과(석사)
1989년 고려대학교 수학과(박사)
1989년~1999년 ETRI 선임 연구원
1999년~2001년 고려대학교 자연과학대학 조교수
2001년~현 재 고려대학교 정보보호대학원 교수
2008년~현 재 고려대학교 디지털포렌식연구센터 센터장
관심분야 : 디지털 포렌식, 심층 암호, 해쉬 함수