포렌식 관점에서 효율적인 파일 카빙 알고리즘 설계 제안*

*권태석 *변근덕 *이상진 *임종인
고려대학교 정보경영공학전문대학원
*kts2341@korea.ac.kr

Design of an efficient file carving algorithm in a forensic perspective

* TaeSuk Kwon *KeunDuck Byun *Sangjin Lee *Jongin Lim
*Graduate School of Information Management and Security, Korea University.

요약

컴퓨터 포렌식 수사에서 파일 복구는 증거 획득에 중요한 요소이다. 하지만 증거 획득 단계에서 하드 디스크와 같은 저장 매체가 손상되거나 파일이 삭제 된 경우가 많이 존재한다. 이 경우 사용할 수 있는 방법 중에 한 가지가 파일 카빙이다. 파일 카빙은 컨텐츠를 알려주는 메타데이터가 손상되거나 없을 때 컨텐츠 및 파일 시그너처를 토대로 파일을 재구성 하는 방법이다. 현재 여러 카빙 알고리즘이 제시되고, 도구들이 나와 있지만 파일이 여러 조각 나 있거나 파일 구조의 Header와 Footer 정보가 없다면 파일 카빙이 힘든 문제점이 있다. 그래서 본 논문에서는 현재의 카빙 알고리즘 및 도구보다 더 넓은 범위를 수용할 수 있는 카빙 알고리즘 설계 방안을 제시 한다.

1. 서론

컴퓨터 기술이 발전하면서 고도화된 디지털 범죄가 등장하기 시작했다. 이러한 범죄에 대응하기 위해서 디지털 증거를 조사하는 기술도 함께 발전되고 있다.

일반적으로 디지털 범죄에 대하여 증거 수집 과정을 거쳐 증거 복구 과정을 수행하며 이러한 데이터를 이용하여 증거분석을 수행한다. 획득한 증거 데이터들은 법정에서 증거로 인정받을 수 있게 된다.

증거데이터 복구 중 파일 복구는 컴퓨터 포렌식 과정에서 증거 획득의 중요한 부분으로 특히 파일 카빙은 파일 복구를 할 때 큰 도움을 줄 수 있는 방법이다. '파일 카빙'이란 파일 시스템 정보나 메타데이터 없이 시그너처와 컨텐츠를 기반으로 파일을 원상태로 복구시키는 방법이다. 증거 획득 시 저장매체의 파일이 손상되거나 삭제되어서 적용할 수 있는 파일 정보가 존재하지 않을 때 파일 카빙을 많이 사용하게 된다.

현재 파일 카빙을 하기 위한 도구는 많이 연구 중이고 여러 카빙 도구들이 나와 있다. 현재 나와 있는 파일 카빙 도구들은 파일 시스템에 상관없이 파일 카빙은 가능하지만 파일이 여러 조각으로 조각나 있거나 파일 구조의 Header나 Footer가 존재하지 않으면 파일 복구율이 매우 떨어진다.

현재 잘 알려진 카빙 도구는 Foremost[1]와 Scalpel[2]로 파일 시스템에 상관없이 파일이 연속적이면서 Header와 Footer정보가 존재하는 경우에 정확한 파일 카빙이 가능하다. 만약 앞에서 말한 정보가 불충분하다면 Foremost와 Scalpel을 통해 파일을 원상태로 복구 시키는 것은 불가능하게 된다. 포렌식 수사에서는 여러 조각난 파일을 복구시켜야 하는 경우가 많이 있기 때문에 파일이 여러 조각나거나 Header나 Footer정보가 없는 경우에 현재의 파일 카빙 기법보다 정확히 적용할 수 있는 파일 카빙 방법이 필요하다.

* 본 연구는 과학재단 디지털 정보 획득 기반 기술 연구 (M10640010005-06N4001-00500)의 지원으로 수행 되었습니다.

본 논문에서는 현재의 파일 카빙 방법의 문제점을 고려하여 파일 구조 정보가 없는 경우와 조각난 파일도 카빙 할 수 있는 파일 카빙 알고리즘 설계 방안을 제안하고자 한다.

2. 관련 연구

가. 파일 시스템

'파일시스템(filesystem)'이란 운영체제가 파티션이나 디스크에 파일들이 연속되게 하기 위해 사용하는 방법이며 자료 구조이다. 즉, 파일들이 디스크 상에서 구성되는 방식이다. 이러한 파일 시스템의 대표적인 예로 FAT와 NTFS가 있다.

- FAT(File Allocation Table)

FAT는 운영체제가 하드디스크 내에 유지하는 일종의 파일 배치 표로써 파일들이 저장되어 있는 클러스터들의 위치도를 제공한다. FAT의 구조는 (그림 1)과 같다.

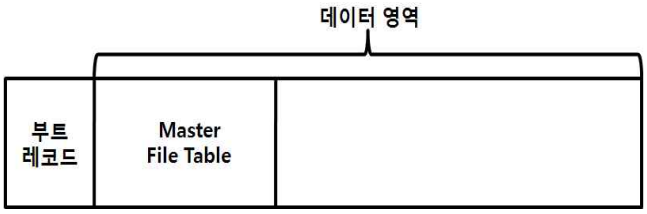
(그림 1) FAT 구조

FAT는 크게 6개 영역으로 나눌 수 있다. 첫 번째 부트레코드 영역에는 윈도우즈를 부팅시키기 위한 기계어 코드와 FAT 파일 시스템의 여러 설정 값들이 담겨 있다. 두 번째 예약된 영역은 나중을 위해 예약해 놓은 영역이다. FAT16인 경우에는 16섹터, FAT32인 경우에는 32섹터를 할당한다. 세 번째 FAT 영역은 클러스터들을 관리하는 테이블이 모여 있는 공간이다. FAT 영역을 통해서 어떤 클러스터가

비어 있는지, 어떤 파일에 어떤 클러스터가 연결되어 있는지 알 수 있게 된다. FAT영역은 각각의 파일이나 디렉토리가 데이터 영역 내에 저장된 위치를 단일 링크드 리스트로 표현하고 있다. 각각의 FAT 엔트리들은 자신의 다음 클러스터(Next Cluster) 값을 담고 있게 된다. 그리고 루트 디렉토리 영역은 FAT16에서 위치가 FAT영역 뒤쪽으로 고정되어 있다. 데이터 영역에는 파일 또는 디렉토리가 저장되어 있고, 마지막 사용하지 않는 영역은 FAT 파일시스템이 볼륨을 구조화시키는 과정에서 잉여분이 조금 남는 영역이다.

- NTFS

NTFS는 윈도우NT 운영체제가 하드디스크 상에 파일들을 저장하고 검색하는데 사용하는 시스템이다. NTFS는 FAT에 비해 성능이나 확장성 및 보안성 면에 있어 많은 개선점들을 제공한다. NTFS의 구조는 (그림 2)와 같다.



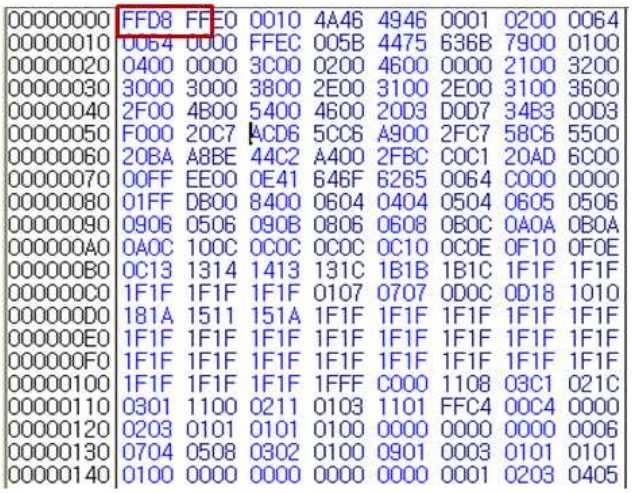
(그림 2) NTFS 구조

NTFS는 크게 부트 레코드와 데이터영역으로 나눌 수 있다. 부트 레코드영역은 FAT 파일시스템의 부트 레코드 구조와 매우 비슷하다. 윈도우즈를 부팅시키기 위한 기계어 코드와 NTFS의 여러 설정 값들이 있으며 볼륨의 크기, 클러스터의 크기, MFT의 시작 주소와 같은 중요한 정보를 저장하고 있다. 데이터 영역에는 MFT(Master File Table)영역이 포함되어 있다. MFT는 볼륨에 존재하는 모든 파일과 디렉토리에 대한 정보를 담고 있는 테이블이다. 파일과 디렉토리 이름, 생성시간, 파일크기, 소유주가 MFT안에 포함 되어 있다. 그리고 MFT는 수많은 MFT 엔트리라는 자료들의 집합으로 이루어져 있고, 하나의 MFT 엔트리는 하나의 파일 또는 디렉터리에 대한 정보를 담고 있다. 0번부터 15번까지 총 16개의 MFT Entry는 파일시스템 관리 데이터를 담고 있는 메타 데이터 파일이 예약되어 있다. 그래서 일반 사용자 파일은 담을 수 없다. 그리고 나머지 데이터영역은 파일과 디렉토리를 담는 영역이다.

나. 파일 시그너처

파일 시그너처는 파일의 내용을 확인하거나 검증해주기 위한 데이터이다. 파일마다 같은 위치에 동일한 시그너처를 가지고 있다면 그 파일의 특징이 될 수 있다. 그래서 이러한 정보를 통해 어떠한 확장자를 가진 파일인지 확인할 수 있게 된다. 예를 들어 JPEG[3]과 MS Compound Document File[4]은 각 Header 부분에 공통된 시그너처를 가지고 있다. 따라서 이 시그너처를 보고 어떤 형태의 파일인지 알 수 있게 되는 것이다.

(그림 3)은 JPEG 파일의 헤더 부분을 보여준다. 대부분의 JPEG 파일의 시작은 0xFF 0xD8 0xFF로 시작해서 0xFF 0xD9로 끝이 난다. JPEG은 즉 '0xFF, 0xD8, 0xFF'와 '0xFF, 0xD9'는 JPEG의 Header와 Footer로 볼 수 있다. MS Compound Document File은 CDH(Compound Document Header)를 첫 번째 섹터에 포함하고 있다. 그리고 CDH의 29번째에는 0xFE와 30번째에는 0xFF를 포함하고 있다. 이러한 시그니처는 MS Compound Document File의 Header부분이라는 것을 알려준다.



(그림 3) JPEG Header

(그림 4) CDH

다. 파일 카빙

위와 같이 파일 시스템이 메타데이터를 기반으로 데이터를 관리하기 때문에 이러한 메타데이터가 손상되거나 삭제되었을 겨우 올바른 데이터를 찾기 불가능하다. AccessData社의 FTK[5]나 Guidance Software社의 EnCase[6]와 같은 파일 복구 도구들도 삭제된 파일의 파일 시스템 내에 남아 있는 메타데이터를 기반으로 파일을 복구한다. 하지만 파일 시스템의 메타데이터 영역이 완전 삭제되었을 경우 기존의 복구 방법으로는 파일을 복구할 수 없다. 따라서 포렌식 수사과정에서 파일 카빙 기술은 메타데이터가 없을 때 파일 복구에 사용할 수 있기 때문에 반드시 필요하다. [표 1]은 현재 가장 널리 사용되는 카빙 도구인 Scalpel과 Foremost에 대해 나타내고 있다.

	Foremost	Scalpel
사용환경	리눅스	윈도우즈/리눅스
사용정보	Header/Footer	Header/Footer
파일상태	연속일 때 가능	연속일 때 가능
최신버전	1.5.3	1.60

[표 1] Foremost와 Scalpel

```
root@linuxbox:~/TS/foremost-1.5.3#
root@linuxbox:~/TS/foremost-1.5.3#
root@linuxbox:~/TS/foremost-1.5.3#
root@linuxbox:~/TS/foremost-1.5.3#
root@linuxbox:~/TS/foremost-1.5.3#
root@linuxbox:~/TS/foremost-1.5.3#
root@linuxbox:~/TS/foremost-1.5.3# ./foremost -c user.conf dfrws-2006-challenge.raw
Processing: dfrws-2006-challenge.raw
|*|
root@linuxbox:~/TS/foremost-1.5.3#
root@linuxbox:~/TS/foremost-1.5.3#
root@linuxbox:~/TS/foremost-1.5.3#
root@linuxbox:~/TS/foremost-1.5.3#
```

(그림 5) Foremost 1.5.3

```
C:\WINDOWS\system32\cmd.exe
Carve lists built.  Workload:
gif with header "\x47\x49\x46\x38\x37\x61" and footer "\x00\x3b" --> 0 files
gif with header "\x47\x49\x46\x38\x39\x61" and footer "\x00\x3b" --> 0 files
jpg with header "\xff\xd8\xff\xe0\x00\x10" and footer "\xff\xd9" --> 19 files
bmp with header "\x42\x4d\x3f\x3f\x00\x00\x00" and footer "" --> 1 files
doc with header "\xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00" and footer "\xd0\xc
doc with header "\xd0\xcf\x11\xe0\xa1\xb1" and footer "" --> 6 files
pst with header "\x21\x42\x4e\xa5\x6f\xb5\xa6" and footer "" --> 0 files
ost with header "\x21\x42\x44\x4e" and footer "" --> 0 files
dbx with header "\xcf\xad\x12\xfe\xc5\xfd\x74\x6f" and footer "" --> 0 files
idx with header "\x4a\x4d\x46\x39" and footer "" --> 0 files
mbx with header "\x4a\x4d\x46\x36" and footer "" --> 0 files
htm with header "\x3c\x68\x74\x6d\x6c" and footer "\x3c\x2f\x68\x74\x6d\x6c\x3
pdf with header "\x25\x50\x44\x46" and footer "\x25\x45\x4f\x46\x0d" --> 0 fil
pdf with header "\x25\x50\x44\x46" and footer "\x25\x45\x4f\x46\x0a" --> 0 fil
txt with header "\x2d\x2d\x2d\x2d\x2d\x42\x45\x47\x49\x4e\x20\x50\x47\x50" and
zip with header "\x50\x4b\x03\x04" and footer "\x3c\xac" --> 13 files
Carving files from image.
Image file pass 2/2.
dfrws-2006-challenge.raw:  21.0% |**************
OPENING C:\scalpel-1.60\scalpel-output/doc-5-0/00000027.doc
OPENING C:\scalpel-1.60\scalpel-output/doc-5-0/00000026.doc
OPENING C:\scalpel-1.60\scalpel-output/doc-4-0/00000021.doc
OPENING C:\scalpel-1.60\scalpel-output/jpg-2-0/00000006.jpg
OPENING C:\scalpel-1.60\scalpel-output/jpg-2-0/00000005.jpg
CLOSING C:\scalpel-1.60\scalpel-output/jpg-2-0/00000005.jpg
OPENING C:\scalpel-1.60\scalpel-output/jpg-2-0/00000004.jpg
CLOSING C:\scalpel-1.60\scalpel-output/jpg-2-0/00000004.jpg
```

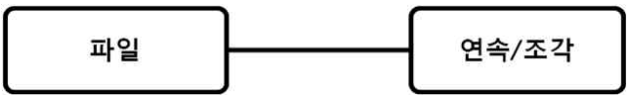
(그림 6) Scalpel 1.60

Scalpel과 Foremost를 이용하여 DFRWS 2006 Challenge[7]에서 제시한 Challenge 파일에서 DOC, HTML, JPEG, ZIP파일을 복구시켜 보았다. 파일을 복구 시킨 결과 두 도구 모두 복구시간은 5초정도로 빠른 편이었고, Foremost가 Scalpel보다 더 많은 파일을 복구시켰다. 하지만 복구가 된 파일의 절반 이상이 제대로 복구되지 않았다. 이러한 결과는 DFRWS 2006 Challenge 파일이 조각 나 있었기 때문이다. 앞의 두 도구는 파일이 조각 나 있거나 Header와 Footer의 정보가 없으면 파일 카빙이 되어도 파일 복구가 원래대로 되지 않는 문제점이 있다는 것을 확인하였다. 따라서 여러 조각난 파일이나 파일 구조 정보를 모르는 경우도 정확히 파일을 카빙 할 수 있는 도구가 필요하다.

3. 개선된 파일 카빙 알고리즘

가. 현재 파일 카빙 알고리즘

현재까지 알려진 파일 카빙 알고리즘은 파일이 연속적이면서, Header와 Footer 정보를 아는 경우에 정확한 파일 카빙이 이루어진다. 파일이 두 조각난 경우의 파일 카빙 알고리즘도 현재 발표되었지만 일부 파일만 가능하고, 아직 도구화 되지는 않아서 좀 더 많은 연구가 필요하다.

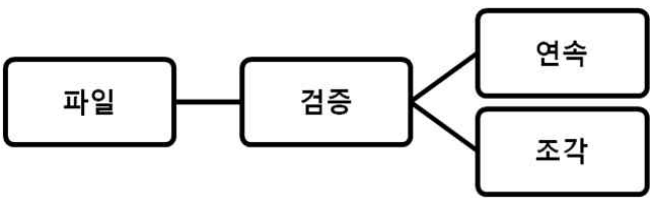


(그림 7) 현재 파일 카빙 알고리즘

(그림 7)에서 보는 것과 같이 현재 파일 카빙 알고리즘의 문제점은 조각난 파일에 상관없이 파일 카빙이 이루어지도록 설계되어 있고, 연속된 파일을 기반으로 파일을 카빙하기 때문에 파일이 조각난 경우 원상태로 파일 복구가 이루어지지 않게 된다. 조각난 파일을 연속된 파일과 구분해 주어서 상태에 맞게 파일 카빙을 도와줄 수 있는 분기점이 필요하다.

나. 개선된 파일 카빙 알고리즘 제안

현재의 파일 카빙 알고리즘은 조각난 파일 구분없이 연속된 파일의 기준으로 파일 카빙을 하기 때문에 원상태로 완벽히 복구 되지 않는 경우가 많았다. 따라서 연속되거나 조각난 파일을 구분해 줄 수 있는 분기점이 필요하다. 이러한 문제점을 해결하기 위해 (그림 8)의 개선된 알고리즘을 제안한다.



(그림 8) 개선된 파일 카빙 알고리즘

제시한 개선된 파일 카빙 알고리즘은 파일을 카빙할 때 먼저 파일이 연속인지 아닌지 검증을 한다. 검증을 하게 되면 연속된 파일인지 조각난 파일인지 확인할 수 있기 때문에 좀 더 정확한 카빙이 이루어질 수 있다. 검증 과정은 두 가지 방법이 있다. Header와 Footer정보, 파일이 포함하고 있는 구조를 통해서 검증할 수 있다. 예를 들어 JPEG은 Header와 Footer가 명확하기 때문에 Header 시그네처가 있는데 Footer 시그네처가 없다면 조각나거나 손상된 파일이라고 판단할 수 있다. 그리고 MS Office 파일은 CDH를 포함하고 있는데 CDH의 알려진 시그네처가 정확하지 않거나 없으면 조각나 있다고 판단할 수 있다. 이 방법으로 검증 오류를 확인해서 손상되거나 분리된 파일인지 구분할 수 있게 된다. 검증이 된 다음에는 파일 상태에 맞게 카빙이 이루어지게 된다. 결국 (그림 8)처럼 분기점을 두어서 파일 상태에 맞는 파일 카빙을 하기 때문에 현재의 파일 카빙 알고리즘보다 좀 더 정확하게

카빙을 할 수 있다.

4. 결론 및 향후 연구 방향

파일 복구는 범죄 상황에서 디지털 증거를 얻기 위한 작업이다. 이러한 디지털 증거 획득을 위해서 사용되는 파일 카빙은 수집한 저장매체가 메타데이터가 손상되거나 삭제되었을 때 파일 복구를 할 수 있는 방법이다. 하지만 현재 나와 있는 카빙 도구 및 알고리즘은 파일이 여러 조각 나 있거나 파일 구조 정보가 없으면 파일 카빙이 제대로 이루어지지 않아 파일을 원상태로 복구하기 어려운 문제점이 있었다. 본 논문에서는 기존의 파일 카빙 도구와 알고리즘을 분석하여 문제점을 제시하였고, 문제점을 바탕으로 조각난 파일과 파일 구조 정보가 없는 경우에 사용할 수 있는 파일 카빙 알고리즘을 제시하고, 기존의 파일 카빙 알고리즘과 비교하여 장점을 제시하였다.

국내의 경우 아직까지 파일 카빙 도구가 나와 있지 않기 때문에 증거 수집의 어려움이 있다. 앞으로 본 논문과 같은 연구를 통해 국내에서도 파일 카빙 도구를 개발하는 노력이 필요하다.

참고 문헌

[1] Foremost 1.3.5, available at
http://foremost.sourceforge.net

[2] Scalpel 1.60 available at
http://www.digitalforensicssolutions.com/Scalpel

[3] JPEG file interchange format v.1.02.
http://www.w3.org/Graphics/JPEG/jfif3.pdf

[4] Microsoft compound document file format v1.3,
http://sc.openoffice.org/compdocfileformat.pdf

[5] Forensic Toolkit(FTK), available at
http://www.accessdata.com

[6] Encase Guidance Software Web Site,
http://guidancesoftware.com
[7] DFRWS 2006 challenge Web Site,
http://www.dfrws.org/2006/challenge

[8] Golden G.Richard III Vassil Roussev, V. Scalpel : A frugal, high-performance file carver. In Proceedings of the 2005 Digital Forensic Research Workshop.

[9] Simon L. Garfinkel, Carving contiguous and fragmented files with fast object validation. In Proceedings of the 2007 Digital Forensic Research Workshop.

[10] Simon L. Garfinkel, Forensic feature extraction and cross-drive analysis. Digit Investing, <http://www.dfrws.org/2006/proceedings/10-Garfinkel.pdf>; August 2006.