안티-포렌식 파일 시스템의 설계

박기현*

Design of Anti-Forensic File System

Park, kihyun

요 약

본 연구는 삭제된 파일을 디지털 포렌식 기술로 복구할 수 없는 파일 시스템의 설계에 대하여 다루고 있다. 이미 삭제된 파일을 손쉽게 복구할 수 있는 기능과 손쉽게 복구할 수 없는 기능의 선택은 전적으로 사용자의 요구에 의하여 결정될 수 있다. 윈도우 시스템과 같이 휴지통 기능을 이용하여 복구를 용이하게 할 수도 있지만, 보안 기능을 강화한다는 측면에서는 복구를 어렵게 할 필요도 있다. UNIX 시스템에서 파일 삭제 기능은 디렉토리 구조에서 삭제할 파일의 연결 고리만을 끊어 주며, 삭제한 파일의 내용 자체를 지우지 않는다. 따라서 디지털 포렌식 기술로 삭제된 파일을 복구하는 것이 가능하다. 본 연구에서 제시한 안티-포렌식 기법은 삭제된 파일의 모든 정보를 의도적으로 임의의 패턴으로 변경함으로써 복구 절차를 무력화 시킨다.

Abstract

This study deals with the design of a file system that cannot recover deleted files with digital forensic technology. The choice of functions that can easily recover already deleted files and functions that cannot be easily recovered can be determined entirely by the user's request. Recycle Bin functions, such as Windows systems, can be used to facilitate recovery, but it is also necessary to make recovery difficult in terms of strengthening security functions. On UNIX systems, the file deletion function only disconnects the files to be deleted from the directory structure and does not erase the contents of the deleted files themselves. Therefore, it is possible to recover deleted files with digital forensics technology. The anti-forensics technique presented in this study disables the recovery process by intentionally changing all information in the deleted file into an arbitrary pattern.

Ⅰ. 서 론

디지털 포렌식은 디지털 장비에 보관된 디지털 정보를 근거로 사실관계를 증명하기 위한 일련의 법률적인 절차와 관계된다[1]. 압수된 디지털 장비에 보관된 데이터의 증거 능력에 관계없이, 본 논문에서는 디지털 장비 사용자가 삭제한 파일을 디지털 포렌식 기술로 복구하는 절차를 무력화시키는 파일 시스템의 설계에 대하여 다룬다[4][16]. 일반적으로 파일 시스템의 보안은 현재 사용 중인 파일을 보호하는 것이 목적이지만[14][15][16], 본 논문은 사용자가 의도적으로 삭제한 파일을 복구할 수 없도록 보호하는 관점에 대하여 다룬다.

* 위덕대학교 경찰정보보안학과 (Department of Cyber Police and Security, Uiduk University)

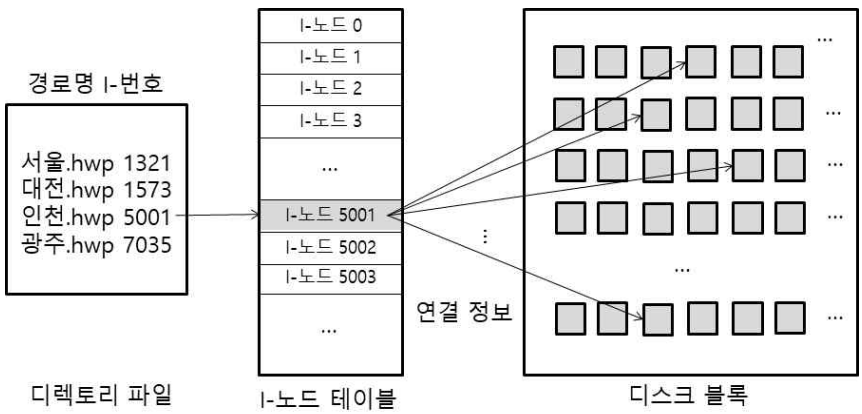
UNIX 시스템의 쉘 환경에서 파일을 삭제하면 즉시 파일이 삭제된다[11]. 그렇지만 그 과정은 파일의 내용 자체가 삭제되는 것이 아니고, 파일의 내용이 보관된 디스크 블록과의 연결 정보를 끊어 버리는 방식이다. 따라서 연결 정보를 복구하는 디지털 포렌식 기술로 삭제된 파일의 내용을 복구하는 것이 가능하다. 이러한 간단한 삭제 기능은 삭제 명령을 빠르게 실행해주는 장점이 있지만, 파일의 내용 자체는 고스란히 하드 디스크에 존재하기 때문에 이론적으로 삭제된 파일의 복구가 가능하다.

본 논문에서는 연결 정보뿐만이 아니고, 파일의 내용 자체를 삭제하는 기능을 보유한 안티-포렌식 파일 시스템의 설계에 관한 내용을 다룬다. 파일의 삭제 과정에서 디스크 블록의 내용을 지우는 추가적인 성능 지연에 관한 선행 연구가 이루어졌으며[7], UNIX 파일 시스템에서 연결 정보인 I-node 구조에서 다루어져야 할 선행 연구가 이루어졌다[3]. 기존 연구에서는 디스크 블록 처리에 따른 성능 저하 문제와 연결 정보의 처리를 등급별로 분석하는 작업이 이루어졌으며, 본 연구에서는 연결 정보에 대한 보다 상세한 설계와 파일 시스템의 디스크 블록 처리 과정에 대한 상세한 설계에 대하여 다룬다.

Ⅱ. 본 론

1. I-노드 테이블과 디스크 블록

파일 시스템은 파일 시스템에 대한 총괄 정보를 보관하는 수퍼 블록과 파일의 내용이 보관될 디스크 블록으로 구분된다[2]. 수퍼 블록에는 파일을 대표하는 I-노드 테이블이 존재하고, 파일을 생성한다는 의미는 I-노드 테이블의 엔트리 하나와 파일 내용을 보관할 디스크 블록을 할당받아 <그림 1>과 같이 연결하면 된다.[2][3][5][6][7][8][9][10]



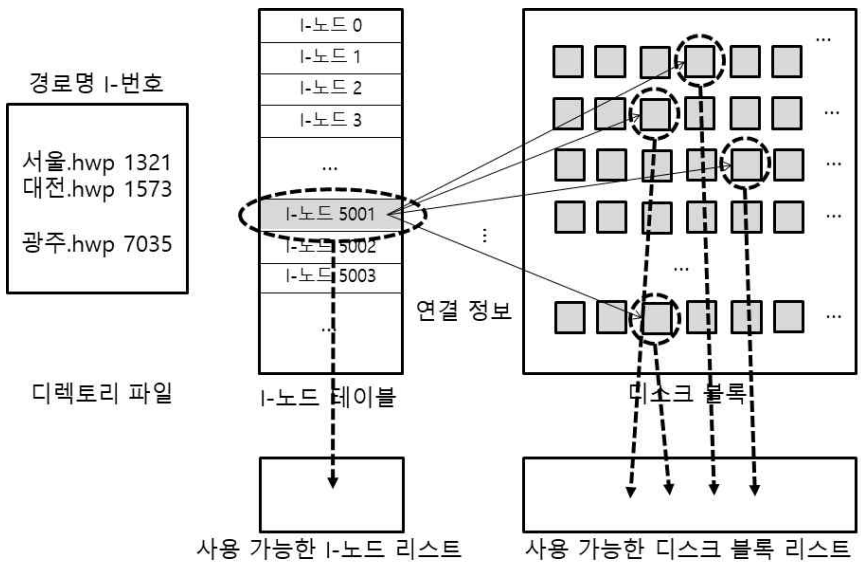
<그림 1. I-노드와 디스크 블록>

디렉토리 파일은 파일의 경로명과 I-노드 번호가 저장되고, 해당 번호에 해당하는 I-노드 테이블의 엔트리에 파일 내용이 보관된 디스크 블록 주소를 포함한 파일의 속성 정보

들이 보관된다. 수퍼 블록에는 사용 가능한 I-노드와 디스크 블록을 관리하기 위하여 다음과 같은 연결 리스크 정보가 포함된다.

- 사용 가능한 I-노드 리스트
- 사용 가능한 I-노드의 수
- 사용 가능한 디스크 블록 리스트
- 사용 가능한 디스크 블록의 수

<그림 2>는 기존 UNIX 시스템의 파일 삭제 과정을 설명한다. 예를 들어 '인천.hwp' 파일을 삭제하면 디렉토리 파일에서 해달 파일에 관한 경로명/I-노드 번호 정보를 지은 후에, 해당 파일에 할당되었던 I-노드 5001번을 '사용 가능한 I-노드 리스트'에 추가하고, 해당 파일의 내용을 보관하던 디스크 블록을 '사용 가능한 디스크 블록 리스트'에 추가하는 것이 전부이다.



<그림 2. UNIX 시스템의 파일 삭제>

<그림 2>에서 I-노드 5001에 기록된 속성 정보, 연결 정보, 디스크 블록의 내용은 그대로 유지된다. 따라서 I-노드 5001의 정보를 복구하거나, 혹은 디스크 블록 주소를 복구하면 이론적으로 삭제된 파일의 내용을 복구할 수 있다[2][3]. 이와 같은 UNIX 시스템의 파일 삭제 과정은 <파일 삭제 알고리즘 1>로 설명된다.

<파일 삭제 알고리즘 1>

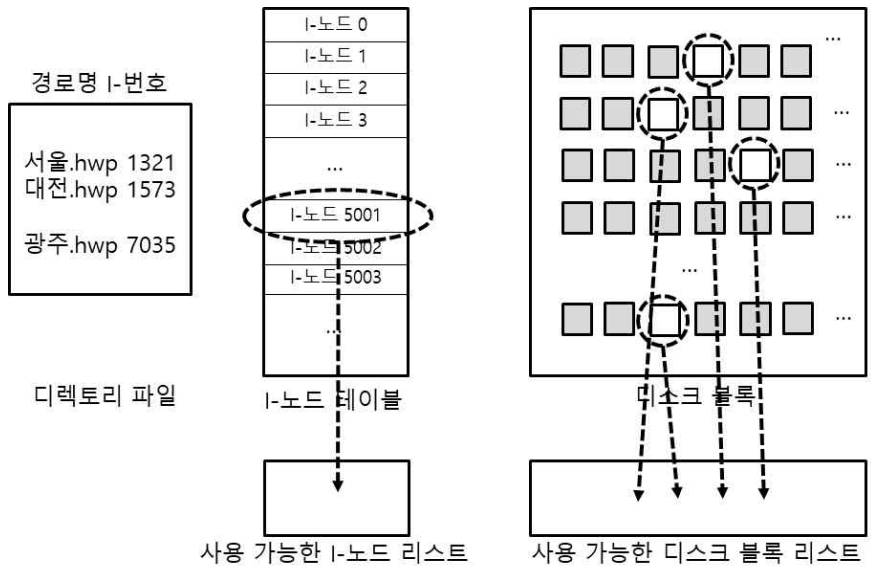
1. inumber = 삭제할 파일의 I-노드 번호

2. inode[inumber]의 모든 디스크 블록을 '사용 가능한 디스크 블록 리스트'로 이동
3. '사용 가능한 디스크 블록 수'의 값 조정
4. inode[number]를 '사용 가능한 I-노드 리스트'로 이동
5. '사용 가능한 I-노드의 수'의 값 조정
6. 디렉토리 파일에서 해당 엔트리를 삭제

이후에 새로운 파일이 생성되면서 삭제된 파일의 I-노드와 디스크 블록이 사용된다면, 그 과정에서 이전에 삭제된 파일의 연결 정보와 디스크 블록의 내용이 변경될 수 있다. 그러나 그전까지는 삭제된 파일의 내용을 디지털 포렌식 기술로 복구하는 것이 가능하다. 본 논문에서 제안하는 안티-포렌식 기법은 '사용 가능한 디스크 블록 리스트'로 이동한 디스크 블록의 데이터를 임의의 패턴으로 완전히 지우는 방법이며, 필요에 따라 연결 정보도 함께 지울 수 있다. 따라서 어떠한 디지털 포렌식 기술로도 삭제된 데이터를 복구하는 것이 가능하지 않다. 또한 I-노드에 보관된 파일의 속성 정보도 제거하여, 해당 파일에 관한 어떠한 정보도 포렌식 기술로 복구할 수도 없도록 설계하였다.

2. 안티-포렌식 알고리즘

<파일 삭제 알고리즘 1>의 문제점은 디스크 블록을 회수하기는 하지만, 그 내용이 그대로 유지된다는 점이다. 본 연구에서 제안한 안티-포렌식 기술이 적용된 파일 시스템에서 연결 정보와 디스크 블록을 완전하게 지우면 <그림 3>과 같이 된다. <그림 2>에서 회색으로 표시된 부분이 <그림 3>에서 하얀색으로 바뀌었다는 의미는 임의의 패턴으로 내용이 지워졌다는 뜻이다.



<그림 3. 안티-포렌식 파일 시스템의 삭제>

이를 지원하기 위해서는 회수되는 디스크 블록을 '사용 가능한 디스크 블록 리스트'로 이동하는 과정에서 그 내용을 임의의 패턴으로 변경해야 한다. 또한 I-노드의 속성 정보마저 지우면 삭제된 파일의 모든 정보가 복구 불가능한 상태가 된다. <파일 삭제 알고리즘 1>을 개선한 안티-포렌식 파일 시스템의 주요 설계 내용에 대하여 살펴본다.

2.1 관련 구조체

파일 시스템의 수퍼 블록에 정의된 구조체에서 다루어지는 주요 필드는 다음과 같다. struct filsys 구조체는 아래의 필드들이 다루어지지만 기존 시스템과 동일한 원리로 동작하며, I-노드를 의미하는 struct dinode 구조체의 필드 중에서 아래에 표기된 필드 값들은 임의의 패턴으로 변경되어 복구할 수 없어야 한다.

```
struct filsys {
        daddr_t s_free[NICFREE]; /*사용 가능한 디스크 블록 리스트 */
        ino_t s_inode[NICINOD]; /* 사용 가능한 I-노드 리스트 */
        daddr_t s_tfree; /* 사용 가능한 디스크 블록의 수 */
        ino_t s_tinoe; /* 사용 가능한 I-노드의 수 */
        ...
}
struct dinode {
        ushort di_uid; /* 사용자 ID이며, 임의의 패턴으로 변경 */
        ushort di_gid; /* 그룹 ID이며, 임의의 패턴으로 변경 */
        off_t di_size; /* 파일 크기이며, 임의의 패턴으로 변경 */
        char di_addr[40]; /* 디스크 블록 주소이며, 임의의 패턴으로 변경 */
        time_t di_atime; /* 마지막 접근 시간이며, 임의의 패턴으로 변경 */
        time_t di_mtime; /* 마지막 변경 시간이며, 임의의 패턴으로 변경 */
        time_t di_ctime; /* 최초 생성 시간이며, 임의의 패턴으로 변경 */
        ...
}
```

2.2 알고리즘

<파일 삭제 알고리즘 1>을 개선하여 안티-포렌식 기법이 적용된 파일 삭제 알고리즘은 다음과 같다. 이 알고리즘의 동작 과정에서는 여러 커널 함수들이 함께 수정되기 때문에 관련 함수를 함께 소개한다. 직접 연결과 1차, 2차, 3차 간접 연결 구조에 관해서는 선행 연구에서 다루었기 때문에 설명을 생략한다[2][3][7][11].

<파일 삭제 알고리즘 2>

1. inumber = 삭제할 파일의 I-노드 번호
2. inode[inumber]와 연결된 모든 디스크 블록의 내용을 임의의 패턴으로 변경
   2.1 직접 1 ~ 직접 10: 파일 내용의 디스크 블록 내용을 지움
   2.2 1차 간접 11: 1차 연결 정보, 파일 내용의 디스크 블록 내용을 지움
   2.3 2차 간접 12: 1차/2차 연결 정보, 파일 내용의 디스크 블록 내용을 지움
   2.4 3차 간접 13: 1차/2차/3차 연결 정보, 파일 내용의 디스크 블록 내용을 지움
3. inode[inumber]의 모든 디스크 블록을 '사용 가능한 디스크 블록 리스트'로 이동
   3.1 직접 1 ~ 직접 10: 디스크 블록 주소
   3.2 1차, 2차, 3차 간접: 연결 정보, 파일 내용의 디스크 블록 주소
4. '사용 가능한 디스크 블록 수'의 값 조정
5. inode[number]의 모든 속성 필드 값을 임의의 패턴으로 변경
6. inode[number]를 '사용 가능한 I-노드 리스트'로 이동
7. '사용 가능한 I-노드의 수'의 값 조정
8. 디렉토리 파일에서 해당 엔트리를 삭제

위의 알고리즘이 동작하는 과정에서 직접적으로 관련이 있는 커널 함수의 변경 사항들은 다음과 같다[2]. 아래에 설명되지 않았지만, 회수되는 디스크 블록을 임의의 패턴으로 지우는 하드 디스크 쓰기 함수에서 임의의 패턴을 생성하는 기능도 추가된다.

- alloc(): 사용할 디스크 블록을 얻는다.
- free(): 삭제할 파일의 디스크 블록을 '사용 가능한 디스크 블록 리스트'로 회수하는 기능이며, 해당 디스크 블록을 회수하기 전에 임의의 패턴으로 변경하는 작업이 선행된다. 이 과정에서 파일의 내용이 보관된 디스크 블록뿐만 아니라, 1차, 2차, 3차 간접의 디스크 블록 주소를 보관하는 디스크 블록의 내용도 지워야 한다.
- ialloc(): I-노드를 얻은 후에, 초기화 기능을 수행한다.
- ifree(): 삭제할 파일의 I-노드를 '사용 가능한 I-노드 리스트'로 회수하는 기능이며, '사용 가능한 I-노드 리스트'로 연결하기 전에 struct dinode 구조체의 I-노드 필드 값을 임의의 패턴으로 지우는 기능이 추가된다.

Ⅲ. 결 론

본 연구에서는 선행 연구인 성능 분석에 대한 검토 결과, 그리고 I-노드 구조와 디스크 블록의 연결 정보에 대한 검토 결과를 바탕으로 디지털 포렌식 기술로 복구할 수 없는 파일 시스템의 설계에 대하여 다루고 있다. UNIX 파일 시스템에서 파일 삭제 기능은 빠른 처리를 위하여 파일의 경로명과 I-노드의 연결 구조만을 끊어 버리는 과정에 불과하다. 이 정도만으로도 일반인들이 삭제된 파일을 복구하는 것은 거의 불가능하다.

파일 시스템의 구조에 대한 이해를 기반으로 하는 디지털 포렌식 기술은 연결 정보를

복구하는 절차, 혹은 연결 정보를 복구할 수 없을 경우는 직접 디스크 블록에 보관된 파일의 내용을 검토하여 삭제된 파일을 복구할 수 있다. 특히 삭제된지 얼마되지 않아서 해당 파일의 디스크 블록들이 다른 새로운 파일의 생성에 사용되지 않은 경우는 파일이 손쉽게 복구될 가능성이 높다. 본 연구에서 설계된 안티-포렌식 파일 시스템은 그와 같은 복구 과정을 어떻게 무력화시킬 수 있는지 설명하고 있다.

본 연구는 UNIX 파일 시스템의 구조에 근거하여 설계가 이루어졌으며, 윈도우 파일 시스템과 같이 그 구조가 다른 파일 시스템에서는 다르게 설계되어야 한다. 이는 디지털 포렌식 기술도 파일 시스템의 종류에 따라 다르게 설계되어야 하는 것과 같은 원리이다.

설계된 파일 시스템은 구현 과정이 복잡하지 않기 때문에 파일 시스템의 종류가 다른 운영체제에서도 쉽게 구현될 수 있으며, 임의의 패턴을 디스크 블록에 저장하는 기술에 대해서만 개별 운영 체제별로 하드 디스크 구동 루틴에 맞추어서 변경해 주어야 한다. 하드 디스크가 존재하지 않는 스마트폰의 경우는 하드 디스크 구동 루틴에 해당하는 메모리 파일 시스템 구동 루틴의 변경이 필요하다.

참고문헌

[1] 이준형, 조정원, “디지털 포렌식의 세계”, 인포더북스, 2013

[2] 방승양, 박찬익 역, “UNIX 운영 체제의 설계”, 대영사, 1998

[3] 박기현, “파일 삭제 등급 분류에 기초한 하드 디스크 보안 시스템의 설계”, 위덕대학교 산업기술연구소 논문집 제17권 제1호, 2013

[4] 이근기, 이창훈, 이상진, “신규 파일 시스템에 대한 디지털 포렌식 분석 필요성 연구”, 한국정보처리학회 추계학술발표대회, 2012

[5] 최진오, “리눅스 환경에서 파일 시스템들의 블록 쓰기 연산 성능 분석”, 한국정보통신학회 논문집 Vol.19 No.1, 2015

[6] Lanyue Lu, Andrea C. Arpaci-Dusseau, Remzi H. Ar, “A Study of Linux File System Evolution, Acm transactions on storage Vol.10 No.1, 2014

[7] 박기현, “삭제 등급이 적용된 보안 파일 시스템의 성능 분석”, 위덕대학교 산업기술연구소 논문집 제20권 제1호, 2016

{8} Tanenbaum, A. S., Herder, J. N., Bos, H., “File Size Distribution on UNIX Systems - Then and Now”, Operating systems review Vol.40 No.1, 2006

[9] 이진호, 손태식, “IOT 플랫폼에 탑재되는 안드로이드 및 리눅스 기반 파일 시스템 포렌식”, 한국디지털콘텐츠학회 논문집 24(2), 2023

[10] 신영훈, 조우연, 손태식, “리눅스 커널에 따른 메타데이터 기반 파일 복원 연구”, 한국정보보호학회 논문집 Vol.29 No.1, 2019

[11] 석상기 역, “UNIX 운영체제”, 사이텍미디어, 1996

[12] 유영준, 고영웅, “커널 레벨 가변 길이 블록 파일 시스템”, 한국정보처리학회, 2016

[13] 안재형, 현철승, 이동희, “비휘발성 메모리 저장 장치를 위한 영속적 페이지 테이블 및

파일 시스템 저널링 기법“, 한국전기전자학회 논문집 Vol.23 No.1, 2019

[14] Zulwaqar Zain Mohtar, Mohd Yazid Idris, Farhan Mohamed, Blockchain-Based Distributed File System Security and Privacy: A Systematic Mapping Study“, 2022 4th International Conference on Smart Sensors and Application(ICSSA), 2022

[15] Jinghan Sun, Shaobo Li, Jun Xu, Jian Huang, “The Security War in File Systems: An Empirical Study from A Vulnerability-Centric Perspective”, ACM Transactions on Storage, 2023

[16] Niusen Chen, Josh Dafoe, Bo Chen, “Poster: Data Recovery from Ransomware Attacks via File System Forensics and Flash Translation Layer Data Extraction”, CCS 22: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security“, 2022