보안공학연구논문지 *Journal of Security Engineering* **Vol.12, No.4 (2015), pp.379-396**
http://dx.doi.org/10.14257/jse.2015.08.02

윈도우즈 파일시스템에서 파일명령 구별을 위한 디지털 포렌식 방법

조규상 1)

ADigitalForensicMethodforaDistinctionofFileCommandsin WindowsFileSystem

Gyu-SangCho[1)]

요 약

이 논문에서는 윈도우즈 NTFS 파일시스템에서 파일명령을 구분할 수 있는 새로운 디지털 포렌식 방법을 제안한다.제안한 방법은 파일명령을 수행할 때 타임스탬프의 변화가 생기는 것을 이용하여 어떤 파일명령이 실행되었는지 구분하는 기능을 수행한다.이것을 구현하기 위하여 TCC & CC(TimestampChangeCheck& CodeConversion)함수,B2D(BinarytoDecimal) 변환함수, FDD(ForensicDecisionDecoder)함수 등의 3 가지 함수를 구현한다.TCC& CC 는 $SI 와 $FN 속성에 들어 있는 각 4 개씩 모두 8 개의 타임스탬프의 변화에 대해 각각 2 비트를 할당하여 전체 16 비트 코드 를 만드는 기능을 하고,B2D 변환 함수는 이것을 10 진수로 변환하여 해당 값을 FDD 함수에 입력하여 디코드된 포렌식을 위한 출력값을 만든다.제안된 방법을 파일생성,파일복사,파일덮어쓰기-소스남김 사례에 적용하여 타임스탬프의 변화에 의해서 어떤 파일명령이 수행되었는지 구분한 결과를 보이기 로 한다.

핵심어 :디지털 포렌식,타임스탬프 변화 패턴,명령어 구분함수,이벤트 재구성,NTFS 파일시스템

Abstract

This research proposes a new digital forensic method for a distinction of file commands in Windows NTFS file system. The proposed method is to distinct what command is executed only by comparing timestamp change pattern before and after a file command execution. The proposed method is composed of three parts, i.e. TCC & CC(Timestamp Change Check & Code Conversion) function, B2D(Binary to Decimal) conversion function, and FDD(Forensic Decision Decoder) function. Each input of the TCC & CC for timestamps of 8 timestamps in $SI and $FN is assigned 2 bits respectively and it produces 16 bit code. The code is converted into a decimal value by the B2D conversion function. The decimal value is decoded into a forensic output by the FDD function. The proposed method gives a forensic way to distinct executed file command. With three forensic cases, i.e. a file creation, a file copy and a file overwrite-a source file left command, the proposed forensic method is verified for its usefulness.

Keywords : Digital forensics, Timestamp change pattern, Command distinction function, NTFS filesystem

접수일(2015 년 07 월 03 일), 심사의뢰일(2015 년 07 월 04 일),
심사완료일(1 차:2015 년 07 월 21 일) 게재확정일(2015 년 08 월 05 일),
게재일(2015 년 08 월 31 일)

[1]750-711 경북 영주시 풍기읍 교촌 1 동양대학교 컴퓨터정보전학과.
email:cho@dyu.ac.kr

*이 논문은 2013 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임.(NRF-2013R1A1A2064426)

ISSN: 1738-7531 JSE

1. 서론

일반적으로 파일 시스템에서 관리하는 파일들에는 작업이 실행된 시간을 기준으로 타임스탬프를 기록한다[1][2].디지털 포렌식의 관점에서는 이런 타임스탬프가 어떤 작업을 언제 했는지 추정하는데 매우 중요한 요소로 작용한다.타임스탬프들을 활용하여 어떤 이벤트를 중심으로 발생된 상황을 추정하여 시간대 별로 재구축하여 포렌식에 활용할 수 있기 때문이다[3].

타임스탬프를 이용한 디지털 포렌식 방법에 관한 연구가 많이 수행되었다.그 중에서 Boyd 와 Poster[4]는 컴퓨터에 기록된 시간이 잘못 다뤄져서 생기는 문제와 마이크로스프트사의 인터넷 익스플로러 웹브라우저에서 지역시간과 UTC 시간 간의 변환 문제,그리고 시간관련 증거를 확증하기 위해 필요한 6 가지 검토사항에 대한 문제를 다뤘다.Stevens[5]는 그의 연구에서 시간 모델을 제시 하였는데 이것은 컴퓨터의 자체 클럭과 여러 전자장비의 디지털 시간의 동작에 영향을 줄 수 있는 여러 요소들을 설명한 것이다.이 연구는 NTFS 뿐 아니라 다른 파일시스템에서도 시스템 시간에 발생하는 시간차이 문제에 대해서 적용할 수 있는 방법이다.Willassen[6]은 타임스탬프에 오류가 포함되어서 문제가 야기되는 경우에 대해서 디지털 포렌식을 위한 해결방법을 제시하였다.그는 타임스탬프를 다루는데 있어서 타임스탬프의 조작과 클럭에 오류 또는 부적절한 설정을 하였을 경 우에 대해서 가설기반 조사방법으로 문제를 해결하기 위한 방법을 제안하였다.이 방법은 클럭에 히스토리적인 조절을 위한 클럭 가설을 설정하고 대상 타임스탬프가 이것에 일치하는지 테스트하 는 방식이다.

윈도우즈의 NTFS 파일시스템에서의 타임스탬프의 분석에 관한 연구사례 중에서 Chow[3]등은 MAC 시간의 발생 순서를 근거로 파일 또는 폴더의 복사,이동,갱신 등의 동작을 추론할 수 있는 7 가지 규칙을 세우고 이것을 파일 생성,복사,압축파일에서 해제,윈도우즈 탐색기 내에서의 프리뷰,폴더 내에서의 개별적인 파일접근 등의 시나리오에 적용한 연구 결과를 제시하였다.이 방법은 일관된 규칙과 논리를 바탕을 근거로 하고 있지는 않지만 NTFS 파일 시스템에서 어떤 이벤트에 대한 타임스탬프의 선후관계를 분석하여 MAC 시간의 동작 특성으로 합리적인 규칙을 제공하고 몇 가지 가설적인 규칙을 세워 실험적으로 증명함으로써 시간 순으로 벌어지는 사건들에 대해 타임스탬프를 근거로 신뢰성 있는 분석을 할 수 있게 되었음에 의미가 있다.

김[7]등은 WindowsNTFS 의 $LogFile 에 들어 있는 로그레코드를 역공학적인 방법으로 로그레코드의 구조를 분석하였고 조[8]는 로그레코드에 기록된 내용이 어떤 파일과 연관된 것이지를 분석
한 연구를 수행하였는데 이 연구들은 파일 작업이 남긴 많은 NTFS 의 로그 레코드들을 포렌식 활용할 수 있는 근거를 마련하였다.

이 연구를 바탕으로 Cho[9]는 파일명령마다 타임스탬프의 변화 패턴을 분석하여 각 명령마다 상이한 패턴을 갖는다는 점을 이용하여 포렌식 분석에 적용할 수 있는 방법에 대한 연구를 수행하 였다.NTFS 의 MFTEntry 의 속성인 $STANDARD_INFORMATION 속성과 $FILE_NAME 속성에 들

어있는 생성시간,수정시간,MFTEntry 수정시간,접근시간 등의 타임스탬프에서의 변화가 생성, 복사,삭제,갱신,이동,이름 변경 등의 파일연산에 따라 서로 상이한 패턴으로 분류가 된다는 것을 조사하였고,타임스탬프 도구를 사용하여 타임스탬프를 변경과 조작한 사례를 탐지하였다.이 연구의 후속 연구[10]에서는 폴더에 대한 타임스탬프의 변화패턴을 분석하고 그것을 제 1 평가함수와 제 2 평가함수의 두 가지 평가함수를 사용하여 포렌식에 활용하는 방법이다.이 평가 함수들의 입력에 타임스탬프 값을 인가하여 파일이나 폴더에 어떤 연산이 수행되었는지를 판단한다.최근의 연구[11]에서는 B-tree 구조를 사용하는 윈도우즈의 디렉토리 인덱스에서 파일이 생성,삭제 등의 각종 연산에 따라 인덱스 목록들에 생기는 변화의 흔적에서 얻을 수 있는 포렌식 정보에 관한 연구를 수행하였다.

이논문에서는 파일 명령을 실행한 후에 타임스탬프의 변화 패턴을 이용하여 어떤 파일 명령이 수행되었는지 판별하여 그것을 포렌식의 증거로 사용하기 위한 방법을 제안한다.이 방법은 파일 명령 실행 전의 타임스탬프와 명령 실행 후의 변경된 타임스탬프 정보를 입력으로 사용하여 두 자리의 비트코드로 변환한다.각 두 자리 비트코드는 $SI 의 4 가지와 $FN 의 4 가지 타임스탬프 등 모 두 8 개의 타임스탬프 변화 값에 의한 것이다.이것은 전체 16 비트로 구성되며 이것을 십진수 값으 로 변환한다.이 값은 포렌식 판별을 위한 디코더를 통해서 어떤 명령이 실행되었는지 판별된다.

이 정수 값들은 입력인수에 주어진 패턴에 따라 서로 다른 고유한 값을 갖게 되어서 어떤 명령이 수행되었는지 알 수 있게 된다.

이논문의 2 장에서는 9 가지의 파일 명령이 실행될 때 타임스탬프에 생기는 변화 패턴에 대하여 논하고 3 장에서는 이 연구에서 제안한 타임스탬프의 변화를 이용하여 어떤 파일명령이 수행되었는지 구별할 수 있는 디지털 포렌식 방법을 설명한다.제안한 방법은 TimestampChangeCheck& CodeConversion(TCC & CC)기능,BinarytoDecimalConversion 기능,그리고 Forensic DecisionDecoder 기능의 3 부분으로 구성된 것에 대하여 각 기능별로 설명하기로 한다.4 장에서는

이 연구에서 제안한 방법을 3 가지 사례에 적용하기로 하며 5 장의 결론에서 이 방법의 의미와 함께 추후과제에 대해 언급하기로 한다.

2.타임스탬프 변화 패턴의 분석

11 기호 표기법

이절에서는 논문에서 사용하는 기호들에 대한 의미를 설명한다.그 내용은 다음과 같다[9]

t_{op}:파일명령이 실행된 시간을 의미한다.

t^C,t^W,t^E, t^A:4 가지 위 첨자에서 C 는 생성(Creation),W 는 쓰기(Write),E 는 MFT 엔트리 수정시간(MFTEntrymodification),A 는 접근시간(Access)을 나타낸다.

t_{sre}:소스 파일의 타임스탬프

t_{Sisre}: 소스 파일의 $STANDARD_INFORMATION 속성의 타임스탬프

u:변경되지 않은 타임스탬프의 총칭,각 필드별로는 u^C, u^W, u^E, u^A 로 표기한다.

$Si:$STANDARD_INFORMATION 속성의 약칭

$F□:$FILE_NAME 속성의 약칭

A :델타시간 -이것은 파일의 연산시간에 추가적으로 소요된 파일처리 시간을 의미한다.이 값은 대체로 작아서 무시할 수 있다.이 연구에서는 0 으로 놓는다.

[표 1]명령 실행에 대한 타임스탬프 변화[9]
[Table1]TimestampChangesbyFileCommand[9]

파일명령	타임스탬프의 변화	
(1)생성	$SI($t^C_{op}$, t^W_{op}, t^E_{op}, t^A_{op})	$FN($t^C_{op}$, t^W_{op}, t^E_{op}, t^A_{op})
(2)복사	$SI($t^C_{op}$, t^W_{src}, t^E_{op}, t^A_{op})	$FN($t^C_{op}$, t^W_{op}, t^E_{op}, t^A_{op})
(3)갱신	$SI($U^C$, t^W_{op}, t^E_{op}, U^A)	$FN($U^C$, U^W, U^E, U^A)
(4)이동-드라이브내	$SI($U^C$, U^W, t^E_{op}, U^A)	$FN($t^C_{SIsrc}$, t^W_{SIsrc}, t^E_{SIsrc}, t^A_{SIsrc})
(5)이동-드라이브 밖	$SI($t^C_{src}$, t^W_{src}, t^E_{op}, t^A_{op})	$FN($t^C_{op}$, t^W_{op}, t^E_{op}, t^A_{op})
(6)덮어쓰기-소스남김	$SI($U^C$, t^W_{src}, t^E_{op}, U^A)	$FN($U^C$, U^W, U^E, U^A)
(7)덮어쓰기-소스지움	$SI($t^C_{src}$, t^W_{src}, t^E_{op}, U^A)	$FN($U^C$, U^W, U^E, U^A)
(8)파일명 변경	$SI($U^C$, U^W, t^E_{op}, U^A)	$FN($t^C_{SIsrc}$, t^W_{SIsrc}, t^E_{SIsrc}, t^A_{SIsrc})
(9)파일속성 변경	$SI($U^C$, U^W, t^E_{op}, U^A)	$FN($U^C$, U^W, U^E, U^A)

2.2 파일명령 실행 후의 타임스탬프 변화

이 절에서는 파일 명령실행 전후의 타임스탬프의 변화를 조사한다.이 타임스탬프의 변화에 대 한 조사는 기존의 연구[9]에서 수행된 것을 본 연구에서 인용한다.기존 연구에서는 타임스탬프의 변화 패턴을 이용하여 타임스탬프 조작을 구별하는데 응용하고 있다.본 논문에서는 이 타임스탬 프 변화 패턴의 상이함을 근거로 어떤 파일 명령이 수행되었는지 판별할 수 있는 기능을 구현하여 디지털 포렌식의 방법으로 사용하는 것을 제안한다.

파일연산은 “생성”,“복사”,“갱신”,“이동-드라이브내”,“이동-드라이브 밖”,“덮어쓰기-소스남김”,“덮어쓰기-소스삭제”,“파일명 변경”,“파일속성 변경”등 9 가지 연산으로 구분된다.기존연구에서는 “MSOffice 에서의 갱신”을 다루고 있지만 이것은 파일명령이 아니므로 이 연구에서는 이 명령을 제외한다.이 조사는 WindowsXP 에서 수행된 것이다.

3.디지털 포렌식을 위한 파일명령 구별함수의 설계

3.1 파일명령 구별함수의 개요

많은 파일의 타임스탬프를 일일이 분석가에 의존하여 분석하는 데는 많은 시간이 걸린다.전문적인 지식을 갖춘 많은 인력이 소요가 된다.타임스탬프의 변화 패턴만을 근거로 어떤 명령이 실행되었는지 알 수 있고 그것을 자동화된 방식으로 구현할 수 있다면 포렌식 분석 작업이 매우 수월해 질 것이다.이 논문에서 타임스탬프의 변화패턴을 이용하여 파일에 적용된 명령이 어떤 명령인지 구별할 수 있는 기능을 갖춘 함수를 설계한다.

본 논문에서 제안한 방법은 3 부분으로 구성된다.Timestamp ChangeCheck & Code Conversion(TCC & CC)기능,BinarytoDecimalConversion 기능,그리고 ForensicDecision Decoder 기능으로 구성된다[그림 1 과 2].

TimestampChangeChecker 는 어떤 명령을 실행했을 때 명령 실행 이전의 것과 명령실행 이후 의 것을 비교하여 달라진 타임스탬프의 부분을 체크한다.CodeConversion 은 타임스탬프의 변화 된 형태에 따라서 두 비트의 이진 코드를 할당하는 역할을 한다[그림 1].

BinarytoDecimalConversion 은 타임스탬프 변화 코드를 십진수 값으로 변환하여 세 번째 단 계의 입력으로 작용한다.

ForensicDecisionDecoder 는 입력된 Decimal 값에 따라 출력을 선택하는 기능을 수행한다.입력 값에 따라 정해진 출력을 선택하는 디코더 기능이다.이 과정까지 수행하고 나면 입력된 타임스탬 프 값들을 근거로 어떤 NTFS 의 파일명령이 실행되었는지 구별할 수 있게 된다.

함수의 출력으로 발생된 값은 명령들마다 고유한 정수 값을 가진다.이 값을 근거로 어떤 파일명령이 수행되었는지 알 수 있다.이 함수의 출력을 구별의 근거로 삼기위해서는 중복된 결과를 만들지 않는 논리적인 방법으로 설계되어야 한다.3.2~3.4 절에서 이 부분에 대한 방법을 제시한다.

3.2TimestampChangeCheck& CodeConversionFunction 함수의 정의

파일 시스템에서 실행된 파일 명령어가 어떤 종류인지 분류하기 위해서 첫 단계로 타임스탬프의변화를 2 비트의 코드로 변환하는 기능을 수행한다.파일명령이 수행되기 이전의 타임스탬프와 명령이 수행된 이후의 타임스탬프에 발생된 변화를 2 비트 코드로 변환한다.$SI 의 4 개의 타임스탬 프 즉,□□□ □□□ □□□ □□□□,$FN 의 4 개의 타임스탬프 즉,□□□ □□□ □□□ □□□□□□,모두 8 개의 TCC 함수에 의하

여타임스탬프 마다 각각 2 비트의 코드로 변환되어 전체 변환되는 비트의 수는 16 비트가 된다[그림 1].[그림 1]에서 □□에서 x 의 의미는 C,W,E,A 를 대표하는 표식으로 사용한 것이다.그러므로 □□□□ □□□ □□□□□□□중의 한 가지 또는 □□□□□□□□□□□□□□□□의 한 가지를 말한다.

각 타임스탬프마다 발생하는 변화패턴은 4 종류가 존재한다. 이것들을 각각 2 비트 코드로 할당하 여 코드화하여 표현한다. 그 종류들은 식(2)에서와 같이 □→□□□□□, □□□→□□□□□, □□□□→□□□□□, □□□□□□→□□□□□로 변환된다.

다음은 TCC 함수에 대한 프로토타입, 기능, 인수, 출력에 대한 설명이다.

함수의 프로토타입 :

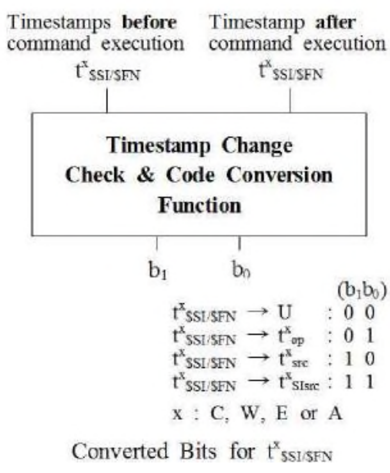
□□□ □□□□□□ □ □□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□ (1)

함수의 기능 : 명령 수행 전의 타임스탬프 값과 명령 수행 후의 타임스탬프 값을 입력으 로 인가하여 비교한 후에 변화된 형태에 따라 2 비트의 코드로 출력하는 기능을 한다.

인수 : 한 개의 TCC 에는 $SI 의 4 개의 타임스탬프 □□□□ □□□□□□ □□□□□□와 $FN 의 4 개의 타임스 탬프 □□□□ □□□□□□ □□□□□□, 즉 전체 8 개의 타임스탬프를 각각 인수로 지정한다. 각 타 임스탬프들은 명령 수행 전의 것과 명령 수행 후의 것을 동시에 입력한다. 전체의 8 개 타임스탬프들은 각각 TCC 의 입력으로 인가된다.

출력 : 함수의 출력 □□□ □□□ □□□(i=0~15 중 짝수)는 입력으로 인가된 타임스탬프의 명령 실행 전의 것과 명령실행 후의 것의 변화에 따라 다음과 같이 두 비트의 값을 출력한다.

$$\begin{aligned} U &\rightarrow (b_{i+1}, b_i)=(0,0) \\ t_{op} &\rightarrow (b_{i+1}, b_i)=(0,1) \\ t_{src} &\rightarrow (b_{i+1}, b_i)=(1,0) \\ t_{SIsrc} &\rightarrow (b_{i+1}, b_i)=(1,1) \end{aligned} \quad (2)$$



[그림 1]TCC& CodeConversion 함수
[Fig.1]TCC& CodeConversionFunction

3.3BinarytoDecimalConversion

8 개의 TCCbox 에서 출력된 16 비트를 B2D(BinarytoDecimal)의 입력으로 인가한다.이 값들은 이진비트이며 B2D 변환기에서 10 진수 값으로 변환한다[그림 2].9 개의 명령 실행 후의 타임스탬프 의 변화를 식(2)에 적용시키고 그것을 B2D 로 변환한 결과는 각각 식(3)-(11)같이 나타낼 수 있다.

[표 1]의 (1)항 “파일의 생성”연산을 적용하여 계산을 하면 다음과 같이 된다.

$$\begin{aligned} O &= f_{B2D}([t^C_{op}, t^W_{op}, t^E_{op}, t^A_{op}]_{\$FN}, [t^C_{op}, t^W_{op}, t^E_{op}, t^A_{op}]_{\$SI}) \\ &= f_{B2D}([(0,1),(0,1),(0,1),(0,1)], [(0,1),(0,1),(0,1),(0,1)]) \\ &= 21,845 \end{aligned} \quad (3)$$

다음은 [표 1]의 (2)항 “파일의 복사”연산을 적용한 결과를 식(4)와 같이 나타낼 수 있다

$$\begin{aligned} O &= f_{B2D}([t^C_{op}, t^W_{op}, t^E_{op}, t^A_{op}]_{\$FN}, [t^C_{op}, t^W_{src}, (t^E_{op}+\Delta), t^A_{op}]_{\$SI}) \\ &= f_{B2D}([(0,1),(0,1),(0,1),(0,1)], [(0,1),(1,0),(0,1),(0,1)]) \\ &= 21,861 \end{aligned} \quad (4)$$

다음은 [표 1]의 (3)항 “파일의 갱신”연산을 적용한 결과이다

$$\begin{aligned} O &= f_{B2D}([U^C, U^W, U^E, U^A]_{\$FN}, [U^C, (t^W_{op}+\Delta), t^E_{op}, U^A]_{\$SI}) \\ &= f_{B2D}([(0,0),(0,0),(0,0),(0,0)], [(0,0),(0,1),(0,1),(0,0)]) \\ &= 20 \end{aligned} \quad (5)$$

다음은 [표 1]의 (4)항 “파일의 이동-드라이브 내”는 같은 드라이브 내에서 파일의 이동 연산이다.이것을 적용한 결과는 다음과 같다.

$$\begin{aligned} O &= f_{B2D}([t^C_{Sfsrc}, t^W_{Sfsrc}, t^E_{Sfsrc}, t^A_{Sfsrc}]_{\$FN}, [U^C, U^W, t^E_{op}, U^A]_{\$SI}) \\ &= f_{B2D}([(1,1),(1,1),(1,1),(1,1)], [(0,0),(0,0),(0,1),(0,0)]) \\ &= 65,284 \end{aligned} \quad (6)$$

다음은 [표 1]의 (5)항 “파일의 이동-드라이브 밖”은 파일이 들어있지 않은 다른 드라이브로 파일을 이동하는 것을 말한다.이것을 적용한 결과는 식 (7)과 같다.

$$O=f_{B2D}(\ [t^{C}_{op},\ t^{W}_{op},\ t^{E}_{op},\ t^{A}_{op}]_{SFN},\ [t^{C}_{src},\ t^{W}_{src},\ (t^{E}_{op}+\Delta),\ t^{A}_{op}]_{SSI}\ )$$
$$=f_{B2D}(\ [(0,1),(0,1),(0,1),(0,1)],\ [(1,0),(1,0),(0,1),(0,1)]\ )$$
$$=21,925 \quad (7)$$

다음은 [표 1]의 (6)항 “파일의 덮어쓰기-소스파일 남김”은 소스 파일을 타겟 파일에 덮어쓰기 하는 것으로 소스파일이 그대로 있는 경우를 말한다.이 연산의 결과는 다음과 같다.

$$O=f_{B2D}(\ [U^{C},\ U^{W},\ U^{E},\ U^{A}]_{SFN},\ [U^{C},\ t^{W}_{src},\ t^{E}_{op},\ U^{A}]_{SSI}\ )$$
$$=f_{B2D}(\ [(0,0),(0,0),(0,0),(0,0)],\ [(0,0),(1,0),(0,1),(0,0)]\ )$$
$$=36 \quad (8)$$

다음은 [표 1]의 (7)항 “파일의 덮어쓰기-소스파일 지움”은 소스 파일을 타겟 파일에 덮어쓰기 하는 것으로 소스파일은 명령실행 후 삭제되는 경우이다.이 경우의 연산의 결과는 다음과 같다.

$$O=f_{B2D}(\ [U^{C},\ U^{W},\ U^{E},\ U^{A}]_{SFN},\ [t^{C}_{src},\ t^{W}_{src},\ t^{E}_{op},\ U^{A}]_{SSI}\ )$$
$$=f_{B2D}(\ [(0,0),(0,0),(0,0),(0,0)],\ [(1,0),(1,0),(0,1),(0,0)]\ )$$
$$=164 \quad (9)$$

다음은 [표 1]의 (8)항 “파일명 변경”연산을 적용한 결과이다

$$O=f_{B2D}(\ [t^{C}_{Slsrc},\ t^{W}_{Slsrc},\ t^{E}_{Slsrc},\ t^{A}_{Slsrc}]_{SFN},\ [U^{C},\ U^{W},\ t^{E}_{op},\ U^{A}]_{SSI}\ )$$
$$=f_{B2D}(\ [(1,1),(1,1),(1,1),(1,1)],\ [(0,0),(0,0),(0,1),(0,0)]$$
$$=65,284 \quad (10)$$

다음은 [표 1]의 (9)항 “파일속성 변경”연산을 적용한 것이다

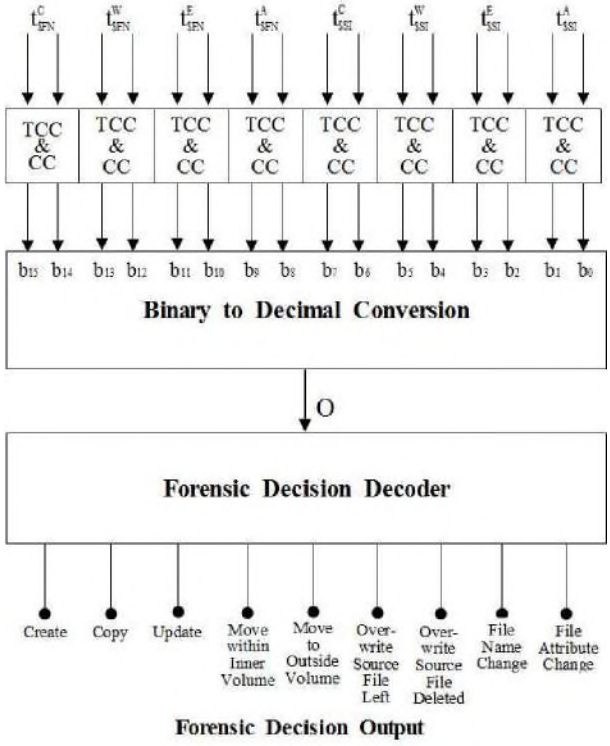
$$O=f_{B2D}(\ [U^{C},\ U^{W},\ U^{E},\ U^{A}]_{SFN},\ U^{C},\ U^{W},\ t^{E}_{op},\ U^{A}]_{SSI}\ )$$
$$=f_{B2D}(\ [(0,0),(0,0),(0,0),(0,0)],\ [(0,0),(0,0),(0,1),(0,0)]\ )$$
$$=4 \quad (11)$$

3.4FDD(ForensicDecision Decoder)의 기능

FDD(ForensicDecisionDecoder)는 B2D 박스[그림 2]에서 출력된 10 진수 값이 함수의 입력으로 사용한다.이 값에 따라 명령어가 분류되어 디코더의 출력으로 선택된다.이 단계에서는 9 가지 명

령 중에서 어떤 파일명령이 수행되었는지 최종적인 판단을 하는 과정이다.

FDD 는 B2D 에서 출력된 10 진수를 3 자리의 이진 코드로 변환하는 작업을 한다.[표 2]는 9 개 명 령의 B2D 의 십진 출력에 해당하는 이진코드를 할당한 표이다.B2D 의 출력 값이 21,845 인 경우는 000 으로 할당하고 디코더를 통해서 파일생성 출력을 선택하여 최종적인 포렌식 판별을 하게 된다 21,861 에는 001 로 코드를 할당하였다.이것이 디코더의 출력으로 파일 복사를 선택한다.출력이 20 인 경우는 010 이 할당되고 디코더가 파일 갱신으로 판별한다.B2D 출력이 65,284 인 경우는 011 로 코드를 할당한다.이것은 드라이브 내에서의 이동의 경우와 파일명 변경의 경우가 동일하게 나타 난다.두 경우를 구분하기 위해서는 타임스탬프의 변화이외의 다른 요소가 필요하다.21,925 의 경 우는 100 으로 코드를 할당한다.디코더를 통해서 드라이브 내의 파일이동으로 판별하게 된다.36 의 경우는 101 로 코드를 할당하고 디코더를 통해서 덮어쓰기-소스남김으로 판별한다.164 의 경우는 110 으로 판별하고 덮어쓰기-소스지움으로 판별한다.4 의 경우는 111 로 코드를 할당하고 이것의 디 코더를 통한 출력은 파일속성 변경으로 판별한다.



[그림 2]파일명령 구별함수의 구조
[Fig.2]Schematicdiagram ofafilecommanddistinctionfunction

[표 2]에서 B2D 에 해당하는 정수 값을 FDD 의 코드 값으로 변환하는 것에 대한 선택은 다양할 수 있다.즉,21845 를 000 으로 할당할 수도 있고,21,861 또는 20 을 000 으로도 할당할 수도 있다.

그 선택은 별로 중요하지 않다.어떤 B2D 의 10 진수에 해당하는 FDD 의 출력의 의미가 올바르게 해석될 수 있으면 어떤 다른 선택에 의하여 만들어진 조합도 가능한 것으로 인정할 수 있다.

[표 2]각 명령에 해당하는 B2D 변환 값
[Table2]B2D ConversionValuesbyFileCommand

B2D 변환값	FDD 코드	FDD 디코더 출력
21,845	000	생성
21,861	001	복사
20	010	갱신
65,284	011	이동-드라이브 내,파일명 변경
21,925	100	이동-드라이브 밖
36	101	덮어쓰기-소스남김
164	110	덮어쓰기-소스지움
65,284	011	파일명 변경
4	111	파일속성 변경

4.사례분석

4.1 작업환경

사례 분석을 위한 작업 환경은 다음과 같이 구성되어 있다

운영체제 :Windows7UltimateKServicePack1
디스크 포맷 :NTFSv3.1
저장매체 :SSD 외장 usb 드라이브
저장공간 :1TB
디스크 할당 클러스터 크기 :4,096 바이트
작업디렉토리명:d:\WorkingDir,d:\SrcDir

4.2 사례 1:파일 생성의 경우

Windows7 에서 d:드라이브에 WorkingDir 디렉토리에 새로운 텍스트 파일을 생성한다.탐색기 창에서 우측마우스를 눌러서 “새로 만들기-텍스트 문서”메뉴를 선택하여 파일을 생성한다.이 파 일은 내용이 들어 있지 않은 빈 파일이다.이 파일에 내용을 기록하게 되면 생성명령 후에 내용수 정 명령이 추가적으로 실행된다.또한 파일명을 “새 텍스트 문서.txt”로 생성한 후에 다른 것으로 변경하게 되면 파일명 변경이 추가적으로 실행된다는 점에 주의해야 한다.

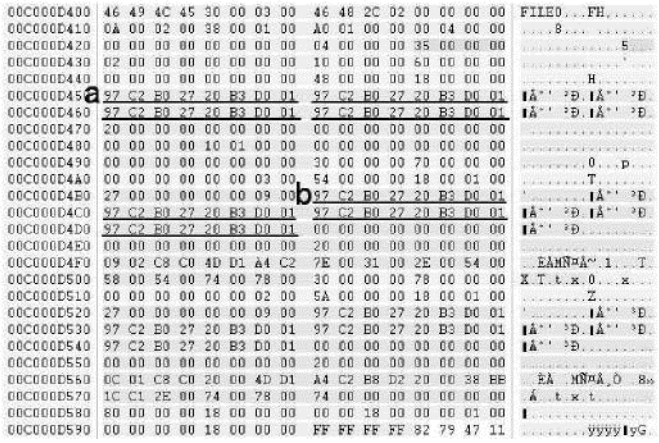
파일연산 :파일생성 -탐색기 창에서 파일생성

생성파일명 :새 텍스트 문서.txt
작업디렉토리:d:\WorkingDir
연산일시 :2015-06-3010:33:07

작업후 파일의 타임스탬프[그림 3]:
새 텍스트 문서.txt
– $STANDARD_INFORMATION
C:2015-06-3010:33:07W:2015-06-3010:33:07
E:2015-06-3010:33:07A:2015-06-3010:33:07
– $FILE_NAME
C:2015-06-3010:33:07W:2015-06-3010:33:07
E:2015-06-3010:33:07A:2015-06-3010:33:07

[그림 3]은 위의 타임스탬프가 MFT 엔트리에 저장되어 있는 모습이다.“a”부분에는 $STANDARD_ INFORMATION 속성의 4 가지 타임스탬프를 나타낸 것이고, “b”부분은 $FILE_NAME 속성의 4 가지 타임스탬프를 나타낸 것이다.각 타임스탬프는 16 진수 8 바이트로 구성 되며 이것은 Windows64 비트 FILETIME 포맷을 사용하고 있다.1601 년 1 월 1 일부터 표시되며 100ns 단위로 표시할 수 있다[12].

TCC&CC 에 입력 타임스탬프를 인가한다.파일 생성의 경우는 과거의 타임스탬프가 존재하지 않 기 때문에 식(12)에는 현재의 타임스탬프만 입력으로 인가하고 과거의 타임스탬프는 입력되지 않 는다.



[그림 3]파일 생성 명령 후의 타임스탬프
[Fig.3]Timestampsafterafilecreationcommand

식(12)을 적용한 후에 각 TCC& CC 기능의 출력은 식(2)에 의하여 식 (13)과 같이 된다.식(13)의 O_{B2D}는 [표 2]의 첫 번째 행에 따라서 식(14)와 같은 결과를 얻는다. $O_{(000),DD}$는 포렌식 디코더 의 출력을 나타내며 식(12)에 주어진 타임스탬프에 의한 것을 포렌식 구별함수에 인가하면 “파일 생성”이라고 구별한다.

$$
\begin{aligned}
O_{TCC}=&[F_{TCC}(0,\ 2015\text{-}06\text{-}30\ \ 10{:}33{:}07),\\
&F_{TCC}(0,\ 2015\text{-}06\text{-}30\ \ 10{:}33{:}07),\\
&F_{TCC}(0,\ 2015\text{-}06\text{-}30\ \ 10{:}33{:}07),\\
&F_{TCC}(0,\ 2015\text{-}06\text{-}30\ \ 10{:}33{:}07)]_{\$FN},\\
&[F_{TCC}(0,\ 2015\text{-}06\text{-}30\ \ 10{:}33{:}07),\\
&F_{TCC}(0,\ 2015\text{-}06\text{-}30\ \ 10{:}33{:}07),\\
&F_{TCC}(0,\ 2015\text{-}06\text{-}30\ \ 10{:}33{:}07),\\
&F_{TCC}(0,\ 2015\text{-}06\text{-}30\ \ 10{:}33{:}07)]_{\$SI}\ )
\end{aligned}
\tag{12}
$$

$$
\begin{aligned}
O_{B2D} &= f_{B2D}(\ [(0,1),(0,1),(0,1),(0,1)]_{\$FN},\ [(0,1),(0,1),(0,1),(0,1)]_{\$SI}\ )\\
&= 21{,}845
\end{aligned}
\tag{13}
$$

$$
O_{FDD} = 21{,}845 \rightarrow (000)_{FDD} : \text{파일생성}
\tag{14}
$$

4.3 사례 2:파일 복사의 경우

d:드라이브의 WorkingDir 디렉토리에서 명령프롬프트에서 copy 명령으로 파일을 복사한다

파일연산 :파일복사 -명령프롬프트에서 copy 명령
원본파일명 :CpSource.jpg
사본파일명 :CpTarget.jpg
작업디렉토리 :d:\WorkingDir
연산일시 :2015-07-0106:21:54

작업후 파일의 타임스탬프 :
CpSource.jpg:[그림 4-(a)]
– $STANDARD_INFORMATION
C:2015-07-0104:08:53W:2015-06-3010:16:36
E:2015-07-0104:08:53A:2015-07-0104:08:53
– $FILE_NAME
C:2015-07-0104:08:53W:2015-07-0104:08:53

E:2015-07-0104:08:53A:2015-07-0104:08:53

CpTarget.jpg:[그림 4-(b)]

– $STANDARD_INFORMATION

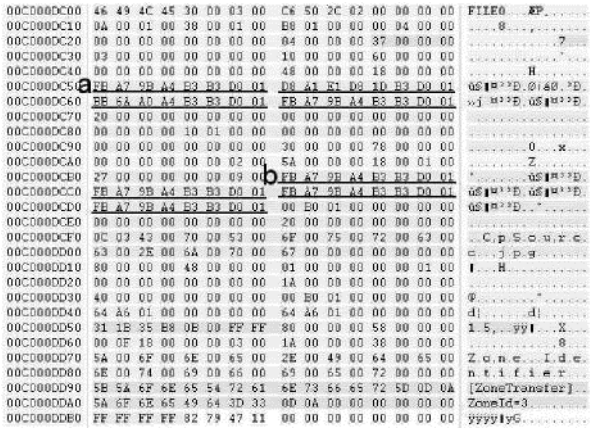
C:2015-07-0106:21:54W:2015-06-3010:16:36

E:2015-07-0106:21:54A:2015-07-0106:21:54

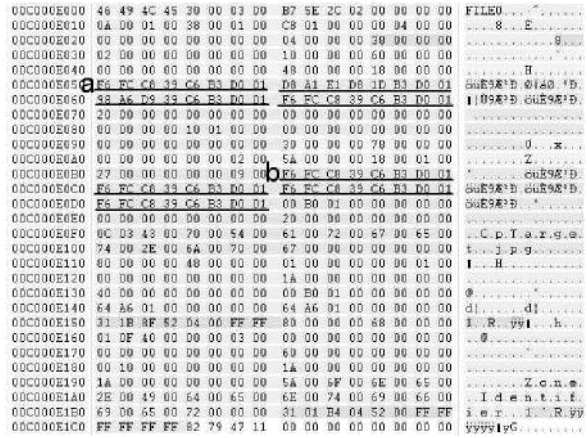
– $FILE_NAME

C:2015-07-0106:21:54W:2015-07-0106:21:54

E:2015-07-0104:08:53A:2015-07-0104:08:53



[그림 4-(a)]복사명령-소스파일의 타임스탬프 [Fig.4-(a)]Filecopycommand– Timestampsforasourcefile



[그림 4-(b)]복사명령-타겟파일의 타임스탬프 [Fig.4-(b)]Filecopycommand– Timestampsforatargefile

[그림 4-(a)]에서 “a”부분에 들어 있는 LittleEndian 방식으로 저장된 타임스탬프 “BB6AA0A4 B3B3D001”생성시간과 접근시간에는 이 타임스탬프는 MFT 엔트리 수정시간 “FBA79BA4B3 B3D001“과 시분초까지 동일하지만 수백 나노초의 시간차이가 있다.파일복사가 수행된 시각과 MFT 엔트리에 기록하는데 미세한 지연시간차를 의미하고 1 초 이하의 시간이므로 복사수행시간에 서 크게 차이가 나지 않는다[12].

TCC&CC 에 입력 타임스탬프를 인가하면 식(15)와 같이 된다

$$
\begin{aligned}
O_{TCC}=[&F_{TCC}(2015\text{-}07\text{-}01\ 04{:}08{:}53,\ 2015\text{-}07\text{-}01\ 06{:}21{:}54),\\
&F_{TCC}(2015\text{-}07\text{-}01\ 04{:}08{:}53,\ 2015\text{-}07\text{-}01\ 06{:}21{:}54),\\
&F_{TCC}(2015\text{-}07\text{-}01\ 04{:}08{:}53,\ 2015\text{-}07\text{-}01\ 06{:}21{:}54),\\
&F_{TCC}(2015\text{-}07\text{-}01\ 04{:}08{:}53,\ 2015\text{-}07\text{-}01\ 06{:}21{:}54)\ ]_{SFN},\\
[&F_{TCC}(2015\text{-}07\text{-}01\ 04{:}08{:}53,\ 2015\text{-}07\text{-}01\ 06{:}21{:}54),\\
&F_{TCC}(2015\text{-}06\text{-}30\ 10{:}16{:}36,\ 2015\text{-}06\text{-}30\ 10{:}16{:}36),\\
&F_{TCC}(2015\text{-}07\text{-}01\ 04{:}08{:}53,\ 2015\text{-}07\text{-}01\ 06{:}21{:}54),\\
&F_{TCC}(2015\text{-}07\text{-}01\ 04{:}08{:}53,\ 2015\text{-}07\text{-}01\ 06{:}21{:}54)\ ]_{SSI}]
\end{aligned}
\tag{15}
$$

식(15)를 적용한 후에 각 TCC& CC 기능의 출력은 식(2)에 의하여 식 (16)과 같이 된다.식(16) 의 OB2D 는 [표 2]의 두 번째 행에 따라서 식(17)과 같은 결과를 얻는다.(001)FDD 는 포렌식 디코 더의 출력을 나타내며 식(15)에서와 같이 주어진 타임스탬프를 포렌식 구별함수에 인가하면 “파일 복사”라고 구별한다.

$$
\begin{aligned}
O_{B2D} &= f_{B2D}([(0,1),(0,1),(0,1),(0,1)]_{SFN},\ [(0,1),(1,0),(0,1),(0,1)]_{SSI}\ )\\
&= 21,861
\end{aligned}
\tag{16}
$$

$$
O_{FDD} = 21,861 \rightarrow (001)_{FDD} : \text{파일복사}
\tag{17}
$$

4.4 사례 3:파일덮어쓰기-소스남김의 경우

d:드라이브의 SrcDir 디렉토리에 들어있는 파일과 WorkingDir 디렉토리에 들어 있는 파일명이 동일한 경우에 파일을 이동하여 덮어쓰기를 하는 경우이다.“덮어쓰기-소스남김”명령은 파일탐색기 창에서 마우스로 파일을 Ctrl+Drag&Drop 을

한다. **파일연산** :파일덮어쓰기-소스남김

명령 **소스파일명** :d:\SrcDir\fstudy.exe

타겟파일명 :d:\WorkingDir\fstudy.exe

연산일시:2015-07-0206:50:16

명령전 파일의 타임스탬프 :

d:\SrcDir\fstudy.exe:[그림 5-(a)]
– $STANDARD_INFORMATION
C:2015-07-0204:44:39W:2014-03-0614:53:26
E:2015-07-0204:51:08A:2015-07-0204:50:54
– $FILE_NAME
C:2015-07-0204:50:54W:2014-03-0614:53:26
E:2015-07-0204:50:54A:2015-07-0204:50:54

d:\WorkingDir\fstudy.exe:[그림 5-(b)]
– $STANDARD_INFORMATION
C:2015-07-0111:45:05W:2015-05-2006:06:25
E:2015-07-0204:47:12A:2015-07-0111:45:05
– $FILE_NAME
C:2015-07-0111:45:05W:2015-06-2903:32:00
E:2015-07-0111:45:05A:2015-07-0111:45:05

명령후 파일의 타임스탬프 :

d:\WorkingDir\fstudy.exe:[그림 5-(c)]
– $STANDARD_INFORMATION
C:2015-07-0111:45:05W:2014-03-0614:53:26
E:2015-07-0206:50:16A:2015-07-0111:45:05
– $FILE_NAME
C:2015-07-0111:45:05W:2015-06-2903:32:00
E:2015-07-0111:45:05A:2015-07-0111:45:05

TCC&CC 에 입력 타임스탬프를 인가하면 식(18)과 같이 되고 식(18)을 적용한 후에 각 TCC&CC 기능의 출력은 식(2)에 의하여 식 (19)와 같이 된다.

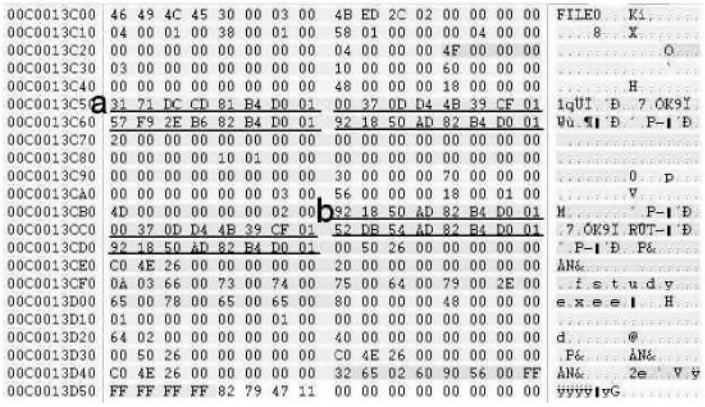
식(19)의 □□□$_D$는 [표 2]의 두 번째 행에 따라서 식(20)과 같은 결과를 얻는다.(□o□)$_{DD}$는 포렌

식 디코더의 출력을 나타내며 식(18)에서와 같이 주어진 타임스탬프를 포렌식 구별함수에 인가하면 “파일덮어쓰기-소스남김”이라고 구별한다.

$$O_{TCC}=[F_{TCC}(2015\text{-}07\text{-}01\ 11{:}45{:}05,\ 2015\text{-}07\text{-}01\ 11{:}45{:}05),\ F_{TCC}(2015\text{-}06\text{-}29\ 03{:}32{:}00,\ 2015\text{-}06\text{-}29\ 03{:}32{:}00),\ F_{TCC}(2015\text{-}07\text{-}01\ 11{:}45{:}05,\ 2015\text{-}07\text{-}01\ 11{:}45{:}05),\ F_{TCC}(2015\text{-}07\text{-}01\ 11{:}45{:}05,\ 2015\text{-}07\text{-}01\ 11{:}45{:}05)\ ]_{\$FN},\ [F_{TCC}(2015\text{-}07\text{-}01\ 11{:}45{:}05,\ 2015\text{-}07\text{-}01\ 11{:}45{:}05),\ F_{TCC}(2015\text{-}05\text{-}20\ 06{:}06{:}25,\ 2014\text{-}03\text{-}06\ 14{:}53{:}26),\ F_{TCC}(2015\text{-}07\text{-}02\ 04{:}47{:}12,\ 2015\text{-}07\text{-}02\ 06{:}50{:}16),\ F_{TCC}(2015\text{-}07\text{-}01\ 11{:}45{:}05,\ 2015\text{-}07\text{-}01\ 11{:}45{:}05)\ ]_{\$SI}] \quad (18)$$

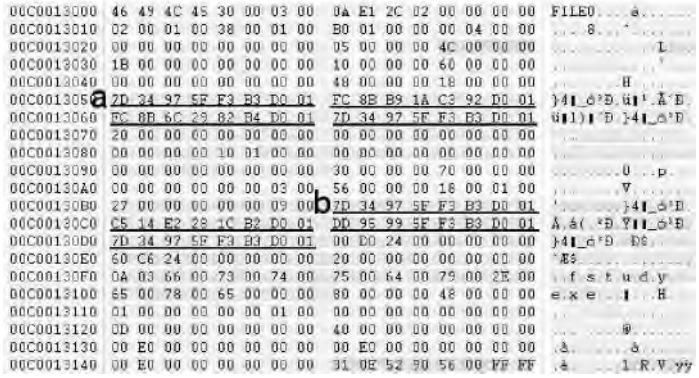
$$O_{B2D} = f_{B2D}([(0,0),(0,0),(0,0),(0,0)]_{\$FN},\ [(0,0),(1,0),(0,1),(0,0)]_{\$SI}\ ) = 36 \quad (19)$$

$$O_{FDD} = 36 \rightarrow (101)_{FDD} : \text{파일덮어쓰기-소스남김} \quad (20)$$



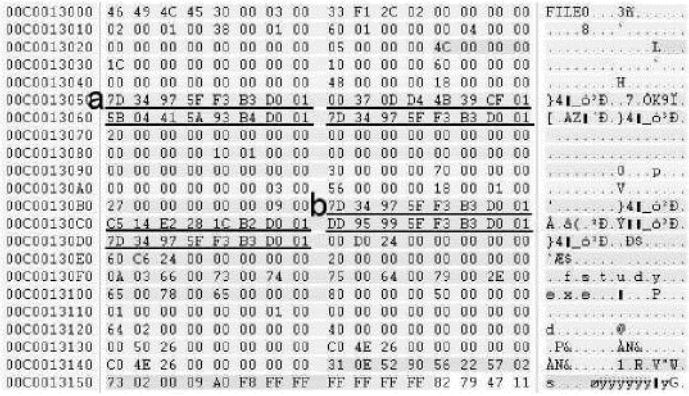
[그림 5-(a)]파일덮어쓰기-원본남김, 소스파일 타임스탬프

[Fig.5-(a)]Fileoverwrite-sourceleft,timestampsforasourcefile



[그림 5-(b)]파일덮어쓰기-원본남김,타겟파일 타임스탬프

[Fig.5-(b)]Fileoverwrite-sourceleft,timestampsforatargetfile



[그림 5-(c)]파일덮어쓰기-원본남김,명령 후 타겟파일 타임스탬프
[Fig.5-(c)]Fileoverwrite-sourceleft,timestampsforatargefileaftercommand

5. 결론

이 연구에서는 파일 명령을 실행한 후에 타임스탬프의 변화 패턴을 이용하여 어떤 파일 명령이 수행되었는지 판별하여 포렌식의 증거로 사용하기 위한 방법을 제안하였다.본 논문에서 제안한 방법은 TimestampChangeCheck& CodeConversion(TCC& CC)기능,BinarytoDecimal Conversion 기능,그리고 ForensicDecisionDecoder 기능 등의 3 부분으로 구성하여 어떤 명령을 실행했을 때 명령 실행 이전과 이후의 달라진 타임스탬프를 비트들로 변환한 후에 십진수 값으로 바꾼 다음 디코더로 어떤 명령이 실행되었는지 구분하는 기능을 수행한다.타임스탬프의 변화를 이용하여 어떤 명령이 수행되었는지 분석할 수 있는 방법이다.이 방법을 구현하는데 있어서 주안 점은 타임스탬프의 고유한 변화 패턴을 중복되지 않는 형태로 파일명령의 실행을 구별할 수 있는 기능을 만들어 내는 것이다.파일의 생성,복사,파일덮어쓰기-원본남김 등의 사례를 통하여 제안된 방법을 적용하여 포렌식을 위한 어떤 명령이 실행되었는지 구별의 과정을 보였다.

[표 1]에 제시된 생성,복사,이동 등의 9 종의 파일 명령의 타임스탬프 변화 패턴에 관한 조사는 선행연구 [9]의 결과를 인용한 것이다.드라이브 내의 이동과 파일명 변경 명령의 타임스탬프 변화 패턴은 같은 패턴이 발생한다.이 경우는 두 명령을 구분하는 추가적인 고려사항이 필요하다.이것은 선행연구 [7-9]에서 $LogFile 을 이용한 방법을 도입하면 어렵지 않게 해결될 수 있을 것으로 생 각된다.

본 연구의 방법은 자동화된 포렌식을 구현하기 위한 목적으로 설계되었다.기존의 포렌식 툴의 경우는 포렌식 분석가의 경험과 직관에 의한 수작업에 많이 의존한다.자동화된 방식은 많은 대상 파일에 적용하여 분석시간을 줄일 수 있다는 의미에서 필요하다.이것을 실현하기 위해서는 완성 도 높은 포렌식 툴을 개발할 필요성이 있다.그 툴을 사용하여 자동적으로 어떤 명령이 파일에 적용되었는지 알 수 있으면 침해사실에 대한 분석이 수월해지기 때문이다.

References

[1] Wikipedia, MAC times, http://en.wikipedia.org/wiki/MAC_times (2015).

[2] B. Carrier, File System Forensic Analysis, Addison-Wesley. (2005), pp. 340-341.

[3] K. P. Chow, Frank Y. W. Law, Michael Y. K. Kwan and K. Y. Lai, The Rules of Time on NTFS File System. SADFE '07, (2007) March, pp. 71-85.

[4] C. Boyd and P. Forster, Time and Date Issues in Forensic Computing - A Case Study. (2004), Digital Investigation, Vol. 1, No. 1, pp. 18-23.

[5] M. W. Stevens, Unification of relative time frames for digital forensics. (2004), Digital Investigation. Vol. 1, No.1 pp. 225-239.

[6] S. Willasen, Hypothesis-based Investigation of Digital Timestamps. IFIP Internation Federation for Information Processing. (2008), Vol. 285, pp. 75-86.

[7] Taehan Kim and Gyu-Sang Cho, A Digital Forensic Method for File Creation using Journal File of NTFS File System. (2010), Journal of KSDIM(ISSN:1738-6667), Vol. 6, No. 2, pp. 107-118.

[8] Gyu-Sang Cho, Method for Finding Related Object File for a Computer Forensics in a Log Record of $LogFile of NTFS File System. (2012), Journal of the Institute of Electronics Eng. of Korea, Vol. 49-CI, No. 4, pp.1~8.

[9] Gyu-Sang Cho, A Computer Forensic Method for Detecting Timestamp Forgery in NTFS. (2013), Computer & Security, Vol. 34, pp. 36-46.9

[10]Gyu-Sang Cho, A Digital Forensic Method by an Evaluation Function Based on Timestamp Changing Patterns. (2014), Journal of KSDIM(ISSN:1738-6667), Vol. 10, No. 2, pp. 91-105.

[11]Gyu-Sang Cho, A Digital Forensic Analysis for Directory in Windows File System. (2015), Journal of KSDIM(ISSN:1738-6667), Vol. 11, No. 2, pp. 73-89.

[12] Microsoft Developer Network, File Times, https://msdn.microsoft.com/en-us/library/ms724290(vs.85,loband).aspx0 (2015).

Author

조규상 (Gyu-SangCho)

1996 년 3 월~현재 :동양대학교 컴퓨터정보전학과 교수

1997 년 2 월 :한양대학교 전자공학과 (공학박사)

2010 년 9 월~2011 년 8 월 미국 Purdue 대학교,Dept.ofComputerInformation Technology,CyberForensicLab,Visitingscholar

관심분야 :디지털 포렌식,시스템보안