Journal of The Korea Society of Computer and Information
Vol. 24 No. 9, pp. 51-58, September 2019
www.ksci.re.kr
https://doi.org/10.9708/jksci.2019.24.09.051

A Digital Forensic Analysis of Timestamp Change Tools for Windows NTFS

Gyu-Sang Cho*

Abstract

Temporal analysis is very useful and important for digital forensics for reconstructing the timeline of digital events. Forgery of a file's timestamp can lead to inconsistencies in the overall temporal relationship, making it difficult to analyze the timeline in reconstructing actions or events and the results of the analysis might not be reliable. The purpose of the timestamp change is to hide the data in a steganographic way, and the other purpose is for anti-forensics. In both cases, the time stamp change tools are requested to use. In this paper, we propose a classification method based on the behavior of the timestamp change tools. The timestamp change tools are categorized three types according to patterns of the changed timestamps after using the tools. By analyzing the changed timestamps, it can be decided what kind of tool is used. And we show that the three types of the patterns are closely related to API functions which are used to develop the tools.

▸Keyword: $STANADARD_INFORMATION Attribute, $FILE_NAME Attribute, Timestamp Change Tool, NTFS Filesystem

I. Introduction

디지털 포렌식의 수행에 있어서 시간정보는 매우 중요하다. 디지털 포렌식 분석작업에서 타임라인을 분석하는 절차는 필수 불가결한 요소이다. 파일의 타임스탬프 위변조는 전반적인 시간적 관계에 불일치를 일으키기 때문에 동작이나 사건들을 재구성하는데 있어서 타임라인 분석에 어려움을 겪게되고 분석의 결과에 대한 신뢰가 떨어질 수 있다[1].

Cho[2]의 연구에서는 NTFS 파일시스템에서 생성, 복사, 갱신, 이동, 덮어쓰기, 파일명변경, 파일속성 변경 등의 파일명령을 수행 전후의 타임스탬프의 변화 패턴을 이용하여 타임스탬프의 조작여부를 알 수 있는 방법을 제안하였다. $LogFile에 기록된 타임스탬프 변경도구를 사용하기 전의 정보와 사용 후의 변경된 정보의 변화패턴에 대한 논리적인 분석으로 타임스탬프에 대한 위변조 여부를 판별하는 방법을 제안하였다.

X. Ding과 H. Zou[3]의 디지털 포렌식 분석에 있어서 증거에 대한 위변조가 중요한 위협인 요소임을 인지하고 이에 대한 대응책으로 신뢰할 만한 시간을 근거로 포렌식을 수행하기 위하여 파일, 디렉토리, 메타데이터들이 갖는 시간적 관계에 적용되는 내정된 규칙을 이용하여 NTFS의 파일시스템에 대한 신뢰할 만한 시간 기반의 포렌식 접근법을 제안하였다. 이것을 위해 파일, 디렉토리, 메타데이터에서 시간값들을 구한 후에, 파일 타입에 근거하여 증거들 중에서 유사하거나 차이나는 것에 대해서 교차 참조를 수행한다. 그리고 타임스탬프에 대한 악의적 접근, 변경, 복사, 위변조와 같은 침해행위에 대하여 식별하는 절차를 진행하는 방식이다.

P. Zdzichowski 등의 연구에서는 안티포렌식에 대한 기술의 전반적인 내용을 설명하고 있는데 그 중에서 타임스탬프의 안티포렌식에 대한 부분에서 timestomp와 SetMACE 도구를 사용하여 타임스탬프에 대한 변경을 수행하는 방법에 대한 설명을 하고 있다. 이 연구에서는 이 도구들을 사용한 후에 남는 증거를 수집하기 위한 방법으로 $LogFile과 $UsnJrnl에 나타난 흔적을

• First Author: Gyu-Sang Cho, Corresponding Author: Gyu-Sang Cho
*Gyu-Sang Cho (cho@dyu.ac.kr), Dept. of Computer, Dongyang University
• Received: 2019. 08. 12, Revised: 2019. 08. 29, Accepted: 2019. 09. 06.
• This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2016R1D1A1B03935646)
• This study was supported by grant from Dong Yang University in 2018

분석하는 과정을 소개하고 있다는 점이 주목할 부분이다[4].

W. Minnaard[5]의 연구에서는 NTFS 파일시스템에서 타임스탬프 변경도구를 사용한 후에 디렉토리 인덱스가 저장된 인덱스 레코드 안의 인덱스 엔트리의 타임스탬프 속성에 기록된 것과 MFT 엔트리에 기록된 것들 간에 일치하지 않는 경우를 찾아서 타임스탬프에 대한 위변조가 이루어졌는지 밝히는 작업을 하였다.

타임스탬프를 안티포렌식을 위한 위변조 목적으로 변경하는 경우도 있지만 스테가노그래픽적인 방법으로 데이터를 숨기기 위한 안티포렌식 방법도 수행되었다. Cho[6]의 연구에서는 NTFS파일시스템에서의 타임스탬프는 100나노초 단위까지 저장하며 1초미만의 시간은 탐색기 창이나 명령프롬프트 창에서 나타나지 않는다는 점에 착안하여 1초미만의 시간이 들어있는 부분에 2바이트의 데이터를 숨기는 방법을 제안하였다. 그는 이어서 2바이트 데이터를 숨기고 남는 비트 부분에 더 많은 정보를 넣을 수 있도록 3바이트 데이터 숨기기 방법에 필요한 확장 비트교정 기법을 제안하였다[7].

같은 시기에 발표된 S. Neuner등[8]의 연구에서는 타임스탬프의 여유공간에 많은 양의 데이터를 저장할 수 있는 방법에 대하여 연구하였다. 이 연구에서 TOMS는 은닉성, 강인성, 넓은 응용성을 제공하는 스테가노그래픽 시스템을 일컫는다. 통상적인 경우에 잘 사용되지 않는 NTFS 타임스탬프의 1초미만의 정보저장를 저장하는 부분에 데이터를 숨기는 기능을 구현하였다. 타임스탬프 속성 중에서 생성시간과 접근시간은 잘 변경되지 않는다는 점을 이용하여 데이터 숨기기 장소로 사용한다. 타임스탬프에 저장하는 부분의 위치는 다르지만 눈에 띄지 않는 타임스탬프의 부분에 데이터를 숨긴다는 기본적인 개념은 Cho[6,7]의 방법과 유사하다.

T. Göbel 등[9]은 S. Neuner등[8]의 NTFS 파일시스템에 수행한 연구에서 착안하여 리눅스의 Ext4 파일시스템에서 같은 방식을 적용한 연구를 수행하였다. 이것은 이전에 발표된 연구[8]를 모체로 하고 있으며 NTFS에서의 방식에서 구현된 것과 같은 방법을 ext4에 적용하여 전문지식이 없는 사용자들이 데이터의 비밀성을 유지하면서 쉽게 사용할 수 있도록 개발하였다.

안티포렌식을 목적으로 타임스탬프를 변경하였거나 또는 데이터 감추기를 위해서 타임스탬프를 변경하였거나 모두 타임스탬프 변경도구를 사용하여 수행한 행위임은 동일하다. 이 논문에서는 이러한 안티포렌식을 목적으로 사용되는 타임스탬프 변경도구들에 대한 기능의 특징들을 디지털 포렌식의 관점에서 분석하기로 한다. 이 논문에서의 기여는 타임스탬프 변경 도구들을 실행한 후에 사용한 도구에 따라 타임스탬프의 변경된 결과에 서로 다른 특징이 나타나는 것을 3가지 타입[19]으로 분류한 것이다. 변경된 타임스탬프를 살펴보면 어떤 도구가 작업에 사용되었는지 알 수 있게 된다. 그리고 이것들은 도구를 개발할 때 사용한 API와 연관성이 있음을 알 수 있게 되므로 안티포렌식을 위해 타임스탬프를 변경하였을 때 사용한 도구를 알 수 있다는 것은 디지털 포렌식에 있어서 매우 중요한 활용방안이다.

이 연구의 2장에서는 NTFS 파일시스템의 타임스탬프의 데이터 구조와 타임스탬프의 시간표현 방법에 대하여 설명한다. 3장에서는 타임스탬프 변경도구들의 동작 특성에 따른 분류와 개발에 사용된 API에 대하여 논하기로 한다. 4장에서는 각 도구들을 사용하여 실험파일의 타임스탬프들을 변경한 결과를 통해서 각 도구들의 특징이 잘 나타난다는 것을 보이기로 한다. 그리고 5장에서 결론으로 마무리 한다.

II. NTFS Timestamps

1. NTFS Timestamp Related Attributes

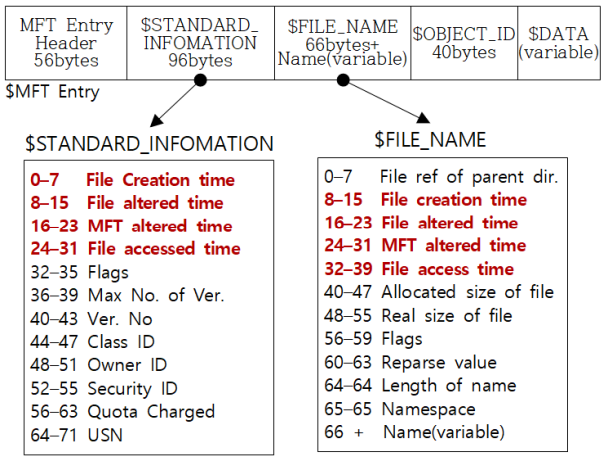

Fig. 1. Structure of $SI and $FN Attributes

Fig. 1은 MFT엔트리에 들어 있는 $STANDARD_INFORMATION속성($SI)과 $FILE_NAME($FN)속성을 나타낸 것이다. $MFT에는 파일의 정보를 갖고 있는 1KB 크기의 MFT엔트리 여러 개로 구성된다. 여기에는 MFT 엔트리 헤더가 제일 먼저온다. 이것의 크기는 56바이트이다. 그 후에 96바이트 크기의 $STANDARD_INFORMATION속성($SI)이 구성된다. 그 다음에는 $FILE_NAME($FN)속성이 구성된다. 이 속성은 파일명을 저장하기 위해서 사용한다. 66바이트의 고정된 데이터 뒤에 가변 길이의 파일명이 66바이트 오프셋 위치에서부터 저장된다. 파일명의 길이는 1~255자까지 가능하다. $FN속성은 8.3포맷의 짧은 파일명을 저장하는 것과 긴 파일명을 저장하는 두 개의 속성을 동시에 가질 수 있다. 이것은 옵션에 따라 선택이 가능하다. 그 뒤에 $OBJECT_ID 속성이 기록되고 파일내용이 저장되는 $DATA 속성이 구성된다. $SI 속성 안에는 4개의 타임스탬프가 들어 있는데 각각 64비트의 생성시간, 수정시간, MFT엔트리 변경시간, 접근시간이 들어 있다. $SI 속성과 마찬가지로 $FN 속성에도 생성시간, 수정시간, MFT엔트리 변경시간, 접근시간 등 4개의 타임스탬프가 들어있다[1,5,6,7]. 이 두 속성의 타임스탬프는 개수와 구성이 동일하지만 실제로 갖고 있는 값은 파일의 연산의 형태에 따라 서로 다른 형태로 저장된다[2].

2. FILETIME

NTFS에서는 시간을 나타내기 위하여 64비트의 FILETIME 데이터 구조를 사용한다. 이 FILETIME 구조는 단일 64 비트 값을 두 개의 32 비트 값 dwLowDateTime(DWORD 타입)와 dwHighDateTime(DWORD 타입)을 결합하여 사용한다[10].

이 방식은 1601년 1월 1일 0시부터 시작한다. 최소단위는 100나노초까지 표현할 수 있다. UTC형식을 채택하고 있어서 지역 시간이 아닌 전세계 표준시간을 저장한다. 시간대(time zone)나 일광절약시간(daylight saving time)을 적용하여도 타임스탬프 자체 값에는 영향을 주지 않는다[10]. FILETIME 데이터 구조에서 저장된 정수값을 사용하여 시간을 표시한다. 이 방식에서는 1초 미만의 값을 표기하기 위하여 10진 정수 7자리를 사용한다. 다시 말하면 정수 1이 100나노초를 나타낸다. 그러므로 1초는 10,000,000가 된다[5,6,7,8].

III. Timestamp Change Tools Classification

1. Type-1: BulkFileChanger etc

Type-1의 도구들은 명령프롬프트 환경에서 실행하는 콘솔형 프로그램과 GUI방식이 지원되는 프로그램들로 구성된다. 대표적으로 FileTouch.exe[20]과 chtime.exe[21]이 있다. GUI 방식으로 구현된 도구들은 SKTimeStamp[22], eXpress TimeStamp Toucher[23], NewFileTime[24], BulkFileChanger[25]과 등의 프로그램이 있다.

Fileapi.h에 속해 있는 아래의 API함수는 인수에 FILETIME 구조체를 사용하여 생성시간, 수정시간, 접근시간 등의 3가지 타임스탬프를 변경할 수 있는 함수이다.

```
BOOL SetFileTime(
  HANDLE hFile,
  CONST FILETIME *lpCreationTime,
  CONST FILETIME *lpLastAccessTime,
  CONST FILETIME *lpLastWriteTime
);
```

SetFileTime() 은 첫 번째 인수에 지정한 파일핸들로 저정된 파일에 생성시간, 수정시간, 접근시간 등의 3가지 타임스탬프를 기록하는 기능을 수행한다[11]. 시간은 초단위까지 지정할 수 있다. MFT 엔트리 수정시간에 대해서는 접근할 수 있는 기능을 갖고 있지 않다. 이 도구를 사용한 순간의 현재 작업시간이 MFT엔트리 수정시간에 기록된다.

Type-1의 도구로 타임스탬프 변경 작업을 수행하고 난 후에는 아래의 예시와 같이 형태의 타임스탬프의 패턴이 나타난다. 숫자의 지정이 불가능한 자리에는 “0”으로 채워진다. 이런 사실은 이미 A. Gungor[12]가 meridiandiscovery.com의 블로그의 문서에서 SKTimeStamp툴을 사용하여 이 현상에 대하여 분석한 바가 있다. MFT엔트리 수정시간에는 작업이 수행된 시간이므로 초미만의 자리에 “0”이 아닌 값들로 채워지는 것이 일반적이다. 각 생성시간, 수정시간, 접근 시간이 모두 “0”으로 채워져 있고 MFT엔트리 수정시간의 초미만이 “0”이 아닌 시간이라면 Type-1의 도구가 사용되었다고 합리적으로 의심할 수 있다.

생성시간: 2019-01-02 12:34:56.0000000
수정시간: 2019-01-02 12:34:56.0000000
MFT엔트리 수정시간: 2019-08-01 11:22:33.7890123
접근시간: 2019-01-02 12:34:56.0000000

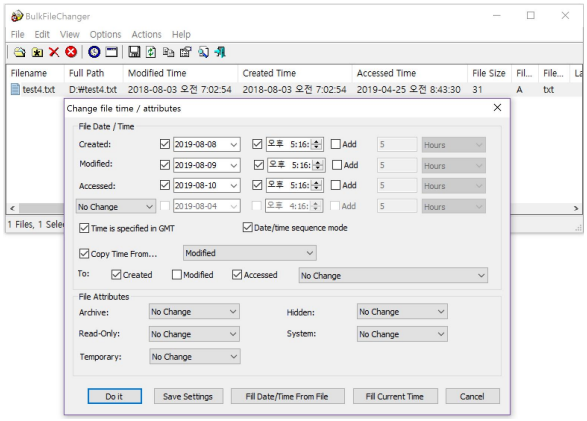

Fig. 2. Screen Shot of BulkFileChanger

2. Type-2: Timestomp

윈도우즈 API에서는 사용자가 시간변경 기능을 제공하고 있다. ZwSetInformationFile()/NtSetInformationFile()함수[13]가 FILE_BASIC_INFORMATION 구조체에는 생성시간, 수정시간, MFT엔트리 수정시간, 접근시간, 파일속성으로 구성되어 있다[14]. 이것은 호출할 때 수정하려는 파일의 파일핸들을 지정하고 변경할 타임스탬프를 적용한다. 모든 타임스탬프를 갱신하지 않고 부분적으로만 갱신할 수도 있다. 이것을 이용하여 2005년 BlackHat 컨퍼런스에서 “timestomp” 안티포렌식을 James Foster와 Vinnie Liu가 제작하여 발표하였다[4,5]. 이 툴은 $SI를 변경하는 기능과 타임스탬프를 삭제하는 기능을 갖고 있다. 이것은 $SI의 4개의 타임스탬프만 직접적으로 변경할 수 있지만 $FN은 변경시킬 수 있는 기능이 없다[6]. 후에 $FN을 수정할 수 있는 우회방법을 알게 되었는데 파일이동 이나 파일명 변경 명령을 수행하면 $SI에 들어있던 타임스탬프 값들이 $FN으로 복사된다는 특징에 착안한 것이다.

독자적인 타임스톰프 유틸리티 개발은 중단되었지만 타임스탬프 변경도구로써 계속적으로 많은 관심을 끌어왔다. 현재 이 툴은 공식적으로 Metasploit Project의 timestomp기능으로 제공되고 있다[15].

timestomp에서 사용하는 시간의 정밀도는 초단위까지만 가능하다. 1초미만의 ms, ns의 시간설정 기능을 갖추고 있지 않다. 이것은 명령프롬프트에서 프로그램을 실행시킬 수 있는데 인수는

Table 1. Timestamp Change Tools and the Characteristics

Types	Tools	$STANDARD_INFORMATION change				$FILE_NAME change				Max Time Precision	Related API	Remarks
		C	M	E	A	C	M	E	A			
1	BulkFileChanger etc	○	○	×	○	×	×	×	×	second	SetFileTime()	Win32 API
2	Timestomp	○	○	○	○	×	×	×	×	second	ZwSetInformationFile()/ NtSetInformationFile()	FILE_BASIC_INFORMATION sturcture
3	SetMACE	○	○	○	○	○	○	○	○	100 nano second	MJ_IRP_WRITE SL_FORCE_DIRECT_WRITE	Kernel Driver Function

3개를 가진다. 첫 번째 인수는 경로를 포함한 파일명을 지정하는데 사용하고 두 번째 인수에는 변경할 타임스탬프의 종류를 지정하는데 사용하는데 쓰기시간에는 -m, 접근시간에는 -a, 생성시간에는 -c, MFT엔트리 수정시간에는 -e를 지정한다. -z의 경우는 4가지 모든 속성 모두 동일한 시간값으로 변경할 때 사용한다. -f는 지정한 파일에 있는 타임스탬프, -v는 타임스탬프 보이기 등의 옵션을 사용할 수 있다. 세 번째 인수에는 “요일 월/일/년 시:분:초 오전/오후"형식으로 날짜와 시간을 지정한다[4,5,7,15].

아래와 같이 d: 드라이브에서 changeTest.txt 파일의 타임스탬프를 -z옵션으로 2019-03-01 12:00:00으로 변경하는 명령을 실행하면 아래와 같이 $SI의 4가지 타임스탬프 모두가 2019-03-01 12:00:00.0000000로 변하게 된다[4]. $FN의 4가지 타임스탬프는 원래 파일이 갖고 있던 시간이 변함없이 그대로 유지된다.

예: d:₩>timestomp.exe d:₩changeTest.txt -z
"Friday 03/01/2019 12:00:00 PM"

$STANDARD_INFORMATION
생성시간: 2019-03-01 12:00:00.0000000
수정시간: 2019-03-01 12:00:00.0000000
MFT엔트리 수정시간: 2019-03-01 12:00:00.0000000
접근시간: 2019-03-01 12:00:00.0000000

3. Type-3: SetMACE

SetMACE[16]는 파일시스템을 우회하여 디스크에 직접 쓰기를 수행하여 타임스탬프를 변경하는 기능을 갖고 있다. 이 방법에서는 $SI와 $FN 모두 동시에 변경가능하기 때문에 기능상 완벽하게 타임스탬프 값을 변경할 수 있다. 그리고 시간의 정밀도는 100ns까지 지정할 수 있어서 NTFS에서 사용하는 타임스탬프의 시간값을 세밀하게 변경할 수 있다. 이것은 파일시스템을 경유하지 않고 물리적인 디스크를 직접 접근하여 타임스탬프에 쓰기 작업을 수행하는 도구[5]라서 기존의 파일시스템관련 API를 사용하는 프로그램이나 NtSetInformationFile()를 사용하는 프로그램과는 근본적으로 동작하는 방식이 다르다.

nt6.x버전이후부터는 디스크 드라이버를 사용자가 직접 접근할 수 없도록 되었다. 만일 그렇게 하려하면 디스크의 마운트를 해제해야하는데 페이지파일을 적용하고 있는 시스템 드라이브의 경우는 마운트를 해제할 수 없기 때문에 다른 방법을 사용하여야 한다[5]. SetMACE에서는 디스크 드라이버에 MJ_IRP_WRITE의 Irp 명령을 보내기 전에 SL_FORCE_DIRECT_WRITE 플래그[17]를 설정하여 이 문제를 해결한 것이다. 이렇게 하면 커널모드 드라이버가 볼륨드라이브의 마운트를 해제하지 않고도 볼륨영역에 쓰기 작업을 수행할 수 있게 된다[16]. 이 기능은 Windows Vista이후 버전부터 가능하다.

SetMACE의 가장 최신 버전은 2014년에 Ver. 1.0.0.16이다. YYYY:MM:DD:HH:MM:SS:MSMSMS:NSNSNSN 형태로 시간을 입력하여 밀리초:나노초 단위까지 직접 변경한다. 타임스탬프 변조도구로는 유일하게 밀리초:나노초를 설정할 수 있다[5,6,7,16].

인수는 네 종류가 사용된다. 첫 번째 인수에는 파일명을 지정하는데 경로를 포함하도록 해야 한다. 두 번째 인수에는 생성시간 "-c", 수정시간 "-m", MFT엔트리 수정시간 "-e", 접근시간 "-a"를 사용한다. 4종류 타임스탬프 모두에 같은 시간값을 지정할 때는 "-z"를 사용한다. "-d"는 기존 타임스탬프를 덤프할 때 사용한다. 세 번째 인수에는 변경하고자 하는 시간을 "2018:08:01:12:34:56:789:1234"의 형태로 지정한다. 네 번째 인수는 변경하려고하는 타임스탬프 속성을 지정하는데 $STANDARD_INFORMATION를 변경하려고 할 때는 “-si", $FILE_NAME를 변경하려고 할 때는 ”-fn"을 지정한다. 둘다 동시에 지정할 때는 “-x”를 사용한다[7,16].

이 도구의 단점은 Windows10에서는 드라이버 로드에러가 나타난다는 것이다. 개발자가 Windows 8버전까지만 테스트하고난 후에 새로운 버전을 등록하지 않고 있어서 Windows10에서는 사용 불가한 상태이다. 이것을 해결하다면 최고의 기능가지고 완벽하게 안티포렌식을 수행할 수 있는 도구가 될 수 있다.

예: d:₩>setmace.exe d:₩changeTest.txt -z
"2019:03:01:12:34:56:789:1234" -x

$STANDARD_INFORMATION
생성시간: 2019-03-01 12:34:56.7891234
수정시간: 2019-03-01 12:34:56.7891234
MFT엔트리 수정시간: 2019-03-01 12:34:56.7891234
접근시간: 2019-03-01 12:34:56.7891234

$FILE_NAME
생성시간: 2019-03-01 12:34:56.7891234
수정시간: 2019-03-01 12:34:56.7891234
MFT엔트리 수정시간: 2019-03-01 12:34:56.7891234
접근시간: 2019-03-01 12:34:56.7891234

IV. Experiments

1. Experimental Environments

이 연구에서 수행하는 타입-1,2,3 도구들에 의하여 수행된 타임스탬프 변경 사례들은 다음과 같은 테스트 환경에서 작업이 이루어진다.

OS : Windows 10 Pro(Ver. 10.0.17134.885)
Windows Vista 8.1(Ver. 6.3.9600)
디스크 포맷 : NTFS v3.1
외장스토리지 드라이브: 2.5“ HDD SCSI
저장공간 : 200기가바이트 파티션(전체 Total 931GB)
디스크 할당 클러스터 : 4K bytes
테스트 파일 : test2.txt, test4.txt, test6.txt
작업디렉토리 : d:₩dirEmpty2
디스크 편집도구 : X-Ways Forensics Ver. 18.4
타임스탬프 표시 변환도구 : TimeLord Ver. 0.1.5.6

2. Experimental Case 1: BulkFileChanger

변경대상의 파일은 test2.txt이다. BulkFileChanger를 사용하여 모든 타임스탬프를 “2022-02-22 22:22:22”로 변경한다. 표 2에는 타임스탬프의 변경전(그림 3)과 변경후(그림 4)의 값들을 기록하였다. 변경후에 나타나는 생성시간, 수정시간, 접근시간이 모두 2022-02-22 22:22:22.0000000로 변경되었다. MFT엔트리 수정시간은 “2019-08-09 15:28:46.7947082”로 기록되었다. 이 시간은 BulkFileChanger를 사용하여 타임스탬프의 변경을 수행한 시간을 의미한다. $FN은 변경되지 않았다.

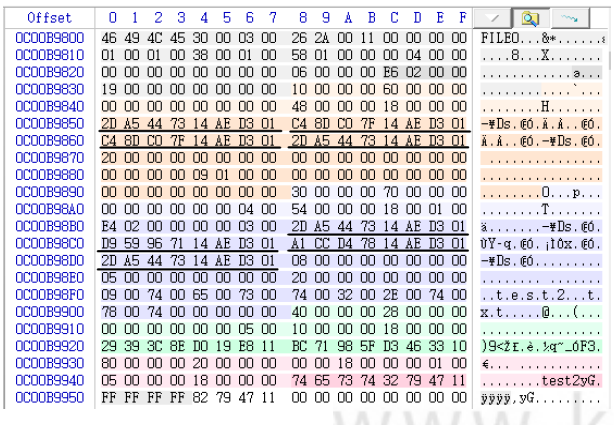

Fig. 3. Screen Shot of Before Using Type-1 Tool

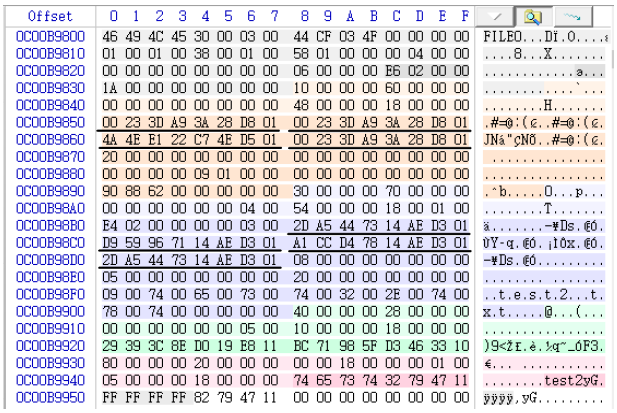

Fig. 4. Screen Shot of After Using Type-1 Tool

Table 2. Before/After Using Type-1 Tool

Attrib.		Before Change	After Change
$SI	C	2DA5447314AED301 2018-02-25 08:41:38.0333869	00233DA93A28D801 2022-02-22 22:22:22.0000000
	M	C48DC07F14AED301 2018-02-25 08:41:58.9780932	00233DA93A28D801 2022-02-22 22:22:22.0000000
	E	C48DC07F14AED301 2018-02-25 08:41:58.9780932	4A4EE122C74ED501 2019-08-09 15:28:46.7947082
	A	2DA5447314AED301 2018-02-25 08:41:38.0333869	00233DA93A28D801 2022-02-22 22:22:22.0000000
$FN	C	2DA5447314AED301 2018-02-25 08:41:38.0333869	2DA5447314AED301 2018-02-25 08:41:38.0333869
	M	D959967114AED301 2018-02-25 08:41:35.2134105	D959967114AED301 2018-02-25 08:41:35.2134105
	E	A1CCD47814AED301 2018-02-25 08:41:47.3667233	A1CCD47814AED301 2018-02-25 08:41:47.3667233
	A	2DA5447314AED301 2018-02-25 08:41:38.0333869	2DA5447314AED301 2018-02-25 08:41:38.0333869

3. Experimental Case 2: Timestomp

변경대상의 파일은 test4.txt이다. 그림 5과 6은 각각 MFT엔트리 내에 들어있는 타임스탬프를 변경전과 변경후의 스크린샷이다.

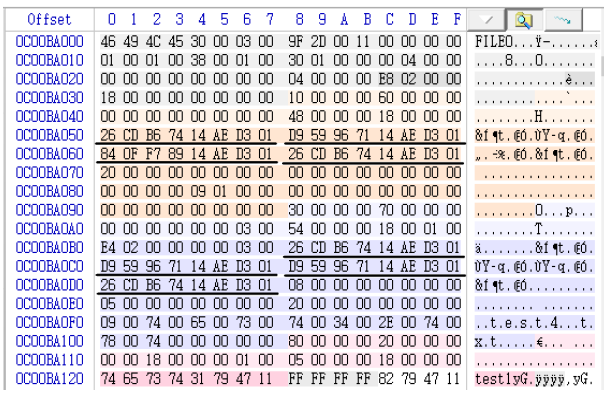

Fig. 5. Screen Shot of Before Using Type-2 Tool

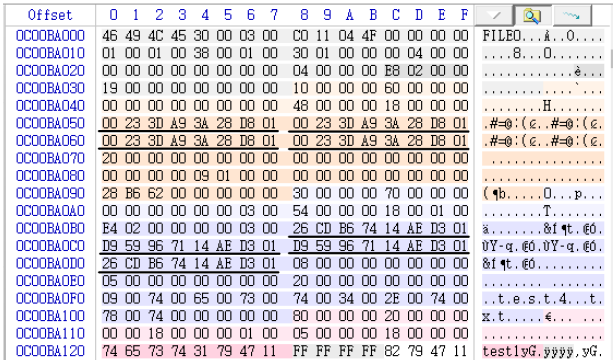

Fig. 6. Screen Shot of After Using Type-2 Tool

Timestomp를 사용하여 모든 타임스탬프를 “2022-02-22 22:22:22”로 변경한다. 그림 5는 변경전의 MFT 엔트리이고, 그림 6은 변경 후의 것을 나타낸 것이다. 표 3에는 타임스탬프의 변경전과 변경후의 값들을 기록하였다. 변경후에 나타나는 생성시간, 수정시간, MFT엔트리 수정시간, 접근시간이 모두 2022-02-22 22:22:22.0000000로 변경되었다. $FN은 변경되지 않았다.

Table 3. Before/After Using Type-2 Tool

Attrib.		Before Change	After Change
$SI	C	26CDB67414AED301 2018-02-25 08:41:40.4592422	00233DA93A28D801 2022-02-22 22:22:22.0000000
	M	D959967114AED301 2018-02-25 08:41:35.2134105	00233DA93A28D801 2022-02-22 22:22:22.0000000
	E	840FF78914AED301 2018-02-25 08:42:16.1125252	00233DA93A28D801 2022-02-22 22:22:22.0000000
	A	26CDB67414AED301 2018-02-25 08:41:40.4592422	00233DA93A28D801 2022-02-22 22:22:22.0000000
$FN	C	26CDB67414AED301 2018-02-25 08:41:40.4592422	26CDB67414AED301 2018-02-25 08:41:40.4592422
	M	D959967114AED301 2018-02-25 08:41:35.2134105	D959967114AED301 2018-02-25 08:41:35.2134105
	E	D959967114AED301 2018-02-25 08:41:35.2134105	D959967114AED301 2018-02-25 08:41:35.2134105
	A	26CDB67414AED301 2018-02-25 08:41:40.4592422	26CDB67414AED301 2018-02-25 08:41:40.4592422

4. Experimental Case 3: SetMACE

변경대상의 파일은 test6.txt이다. 그림 7과 8은 각각 MFT 엔트리 내에 들어있는 타임스탬프를 변경전과 변경후의 스크린샷이다. SetMACE를 사용하여 모든 타임스탬프를 “2022-02-22 22:22:22.2222222”로 변경한다. 표 4에는 타임스탬프의 변경전과 변경후의 값들을 기록하였다. 변경후에 $SI와 $FN의 생성시간, 수정시간, MFT엔트리 수정시간, 접근시간 등 모두 8개의 타임스탬프가 2022-02-22 22:22:22.2222222로 변경되었다. 이 프로그램은 Windows 10에서는 드라이버 로드에러가 발생하기 때문Windows Vista 8.1에서 실험을 수행하였다.

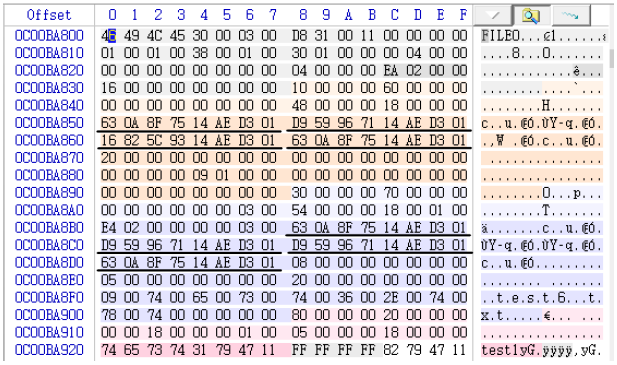

Fig. 7. Screen Shot of Before Using Type-3 Tool

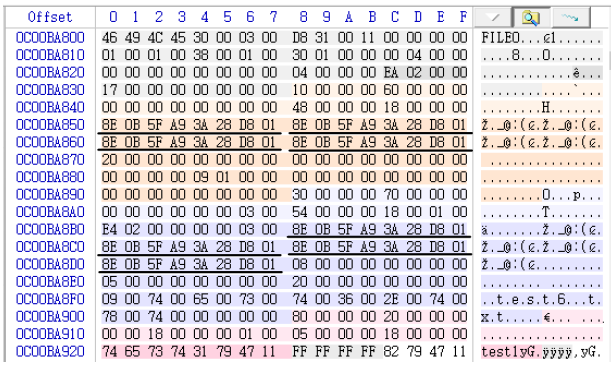

Fig. 8. Screen Shot of After Using Type-3 Tool

Table 4. Before/After Using Type-3 Tool

Attrib.		Before Change	After Change
$SI	C	630A8F7514AED301 2018-02-25 08:41:41.8763875	8E0B5FA93A28D801 2022-02-22 22:22:22.2222222
	M	D959967114AED301 2018-02-25 08:41:35.2134105	8E0B5FA93A28D801 2022-02-22 22:22:22.2222222
	E	16825C9314AED301 2018-02-25 08:42:31.8768662	8E0B5FA93A28D801 2022-02-22 22:22:22.2222222
	A	630A8F7514AED301 2018-02-25 08:41:41.8763875	8E0B5FA93A28D801 2022-02-22 22:22:22.2222222
$FN	C	630A8F7514AED301 2018-02-25 08:41:41.8763875	8E0B5FA93A28D801 2022-02-22 22:22:22.2222222
	M	D959967114AED301 2018-02-25 08:41:35.2134105	8E0B5FA93A28D801 2022-02-22 22:22:22.2222222
	E	D959967114AED301 2018-02-25 08:41:35.2134105	8E0B5FA93A28D801 2022-02-22 22:22:22.2222222
	A	630A8F7514AED301 2018-02-25 08:41:41.8763875	8E0B5FA93A28D801 2022-02-22 22:22:22.2222222

V. Conclusion

이 논문에서는 타임스탬프 변경도루를 사용하였을 때 나타나는 현상을 디지털 포렌식의 관점에서 분석하였다. 타임스탬프 변경도구의 분류의 기준은 사용된 API와 변경시간의 정밀도이다. 도구를 제작하는데 사용한 API에 따라서 변경시간의 정밀도가 결정되기 때문에 두 분류 기준항목은 서로 연관성을 갖는다. 제안한 기준에 따라서 도구들을 3종류로 구분하였다.

Type-1에 해당하는 도구들은 FileTouch.exe, chtime.exe, SKTimeStamp, eXpress TimeStamp Toucher, NewFileTime 등이다. Type-2에는 timestomp가 해당하고, Type-3에는 SetMACE가 분류되었다.

Type-1의 경우는 생성시간, 수정시간, 접근시간 등의 3가지 타임스탬프만 수정이 가능하였고 1초 미만의 값이 모두 0으로 기록되는 "년-월-일 시:분:초.0000000"형태가 되는 특징이 나타났다. 이 경우에 MFT엔트리 수정시간의 경우는 타임스탬프를 변경 작업이 이루어진 그 시간이 기록되는 특징이 나타났다.

Type-2의 경우는 "년-월-일 시:분:초"까지만 수정이 가능하다. "년-월-일 시:분:초.0000000"형태가 되는 특징을 보인다. $FN의 타임스탬프를 변경하려면 timestomp의 기능이 아닌 우회적인 방법으로 윈도우즈의 파일이동 명령을 사용한다.

Type-3의 경우는 "년-월-일 시:분:초.NSNSNSN"형태로 100나노초 단위까지 까지 수정이 가능하다. 현재 제공되고 있는 1.0.0.16버전은 Windows 8에서 테스트되었다고 제작자가 밝히고 있다[16]. Windows 10에서는 드라이버 로드에러로 인해서 실행이 되지 않고 있다.

안티포렌식 수행을 위하여 각 타입의 툴들이 사용되고 난 후에 나타나는 타임스탬프 변경의 특징들에 대하여 살펴보았다. 이 논문에서 제안한 도구들의 분류와 타임스탬프의 데이터 특징을 근거로 타임스탬프가 변경되었을 때 어떤 타입의 도구가 사용되었는지 디지털 포렌식적으로 분석할 수 있다는 의미가 된다. 4장에서 각 3가지 타입에 대하여 실험을 수행하여 각 테스트 파일들의 타임스탬프가 각 타입에 해당하는 형태로 나타나는 것을 결과로 입증하였다.

이 연구의 후속 연구로써 제안한 방법을 3가지 타입에서 보이고 있는 단점들에 대한 보완적 연구가 필요해 보인다. 그 중에서 타입-2와 타입-3 도구의 기능 개선 연구가 필요하다. timestomp의 초미만 시간값 설정과 우회방법이 아닌 직접적인 $FN을 변경할 수 있는 기능을 만드는 연구가 필요하다. 또한, SetMACE의 경우는 Windows 10에서 드라이버 로드에러를 수정하면 현존하는 최고의 타임스탬프 안티포렌식 도구로 사용될 수 있을 것으로 생각된다.

REFERENCES

[1] Sebastian Neuner et. al., "Timestamp hiccups: Detecting manipulated filesystem timestamps on NTFS", Proc. of the 12th Int. Conf. on Availability, Reliability and Security(ARES '17), Aug. 29, 2017.

[2] Gyu-Sang Cho, "A Computer Forensic Method for Detecting Timestamp Forgery in NTFS", Computer & Security, Vol. 34, pp. 36-46, 2013. 3]

[3] X. Ding, H. Zou, "Reliable Time Based Forensics in NTFS", 2010 Annual Computer Security Applications Conference, Dec. 6-10, 2010.

[4] P. Zdzichowski et.al., "Anti-Forensic Study", NATO CCDCOE(NATO Cooperative Cyber Defence Centre of Excellence), www.ccdcoe.org, 2015.

[5] Wicher Minnaard, "Timestomping NTFS," IMSc final research project report, University of Amsterdam, Faculty of Natural Sciences, Mathematics and Computer Science, 2014.

[6] Gyu-Sang Cho, "Data Hiding in NTFS Timestamps for Anti-Forensics", International Journal of Internet, Broadcasting and Communication, vol. 8, no. 3, pp. 31-40, 2016.8

[7] Gyu-Sang Cho, "A Steganographic Data Hiding Method in Timestamps by Bit Correction Technique for Anti-Forensics", Journal of The Korea Society of Computer and Information, Vol. 23 No. 8, pp. 75-84, August 2018.8

[8] Neuner, S. et. al., "Time is on my side: stegano- graphy in filesystem metadata," Digital Investigation, 18, pp. S76-S86. 2016.

[9] T. Göbel and H. Baier, "Anti-forensics in ext4: On secrecy and usability of timestamp-based data hiding," Digital Investigation, 24, pp. S111-S120, 2018.

[10] INFO: Working with the FILETIME Structure, https://support.microsoft.com/en-us/help/188768/info-working-with-the-filetime-structure

[11] Microsoft Windows Dev Center, "SetFileTime function", https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-setfiletime

[12] A. Gungor, "Date Forgery Analysis and Timestamp Resolution", https://www.meridiandiscovery.com/articles/date-forgery-analysis-timestamp-resolution/, August 11, 2014

[13] Microsoft Hardware Dev Center, "NtSetInformation File function", https://docs.microsoft.com/ko-kr/windows-hardware/drivers/ddi/content/ntfs/nf-ntifs-ntsetinformationfile

[14] Microsoft Hardware Dev Center, "FILE_BASIC_INFORMATION sturcture", https://docs.microsoft.

com/en-us/windows-hardware/drivers/ddi/content/wdm/ns-wdm-file_basic_information
[15] Metasploit Anti Forensics Project, http://www.metasploit.com/research/projects/antiforensics
[16] SetMace, “https://github.com/jschicht/SetMace”
[17] Microsoft Hardware Dev Center, “IRP_MJ_WRITE”, https://docs.microsoft.com/ko-kr/windows-hardware/drivers/ifs/irp-mj-write
[18] Ahmed A. Bahjat and Jim Jones, "Deleted file fragment dating by analysis of allocated neighbors", Digital Investigation, Vol.28, pp. S60-S67, 2019.
[19] Gyu-Sang Cho, “Digital Forensic Analysis of Timestamp Change Tools: An Anti-Forensics Perspective”, Proceedings of KSCI Summer Conference 2019 Vol. 27 No. 2, pp. 391-392, July 2019.
[20] FileTouch, “http://www.softtreetech.com/24x7/archive/47.htm”
[21] chtime, “https://github.com/Loadmaster/chtime-win32”
[22] SKTimeStamp, https://tools.stefankueng.com/SKTimeStamp.html
[23] eXpress TimeStamp Toucher, “https://www.softpedia.com/get/System/File-Management/TimeStamp-Toucher.shtml
[24]NewFileTime,“https://www.softwareok.com/?seite=Microsoft/NewFileTime”
[25] Bulk File Changer, “https://www.nirsoft.net/utils/bulk_file_changer.html”

Authors

Gyu-Sang Cho received the B.S., M.S. and Ph.D. degrees in Electronic Engineering from Hanyang University, in 1986, 1989 and 1997, respectively. Dr. Cho joined the faculty of the Department of Computer at Dongyang University, Yeongju, Korea, in 1996.
He is interested in digital forensics, system secirity, and AI security.