사물인터넷융복합논문지 Vol. 9, No. 6, pp. 23-28, 2023 https://doi.org/10.20465/KIOTS.2023.9.6.023

NTFS에서 저장장치 성능을 활용한 타임스탬프 변조 탐지 기법 설계

송종화[1], 이현섭[2*]
[1]성균관대학교 과학수사학과 대학원생, [2]백석대학교 컴퓨터공학부 교수

A Design of Timestamp Manipulation Detection Method using Storage Performance in NTFS

Jong-Hwa Song[1], Hyun-Seob Lee[2*]
[1]Graduate Student, Department of Forensic Sciences, Sungkyunkwan University
[2]Professor, Division of Computer Engineering, Baekseok University

요 약 Windows 운영체제는 다양한 데이터를 타임스탬프와 함께 로깅한다. 타임스탬프 변조는 안티포렌식의 한 행위로 용의자가 범행과 관련된 데이터의 타임스탬프 조작을 통해 흔적을 숨겨 분석관이 사건의 상황 재현을 어렵게 하여 수사를 지연시키거나 중요한 디지털 증거 획득을 실패하게 만든다. 따라서 이를 대처하기 위해 타임스탬프 변조를 탐지하는 여러 기법이 개발되었다. 그러나 만일 용의자가 타임스탬프 패턴을 인지하고 정교하게 시간을 조작하거나 변조 탐지에 활용되는 시스템 아티팩트를 변경한다면 탐지가 어렵다는 한계점을 가지고 있다. 본 논문에서는 용의자가 파일의 타임스탬프를 조작하더라도 저장장치의 속도에 비례하여 1초 미만의 단위값까지를 고려한 정교한 변경이 어려움에 착안하여, 타임스탬프 변조를 탐지할 수 있는 기법을 설계하고자 한다. 설계한 탐지 기법에서는 우선 변조가 의심스러운 파일의 타임스탬프를 확인하여 해당 파일의 쓰기시간을 확인한다. 그다음 확인된 시간을 저장장치의 성능을 고려하여 시간 내에 기록된 파일 크기와 대조한다. 마지막으로 특정 시간에 파일이 쓰인 총용량을 구하고 저장장치의 최대 입출력 성능과 비교하여 파일의 변조 여부를 탐지한다.

주제어 : 타임스탬프, 변조 탐지, 저장장치, NTFS, 안티포렌식

Abstract Windows operating system generates various logs with timestamps. Timestamp tampering is an act of anti-forensics in which a suspect manipulates the timestamps of data related to a crime to conceal traces, making it difficult for analysts to reconstruct the situation of the incident. This can delay investigations or lead to the failure of obtaining crucial digital evidence. Therefore, various techniques have been developed to detect timestamp tampering. However, there is a limitation in detection if a suspect is aware of timestamp patterns and manipulates timestamps skillfully or alters system artifacts used in timestamp tampering detection. In this paper, a method is designed to detect changes in timestamps, even if a suspect alters the timestamp of a file on a storage device, it is challenging to do so with precision beyond millisecond order. In the proposed detection method, the first step involves verifying the timestamp of a file suspected of tampering to determine its write time. Subsequently, the confirmed time is compared with the file size recorded within that time, taking into consideration the performance of the storage device. Finally, the total capacity of files written at a specific time is calculated, and this is compared with the maximum input and output performance of the storage device to detect any potential file tampering.

Key Words : Timestamp, Manipulation Detection, Storage, NTFS, Anti-forensics

본 논문은 2023년도 교육부의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(NRF2021R1I1A3061020)과 지자체-대학 협력기반 지역혁신 사업(2021RIS-004)의 결과입니다.
교신저자 : 이현섭(hyunseob@bu.ac.kr)
접수일 2023년 10월 12일 수정일 2023년 12월 9일 심사완료일 2023년 12월 14일

1. 서론

범죄가 발생할 가능성이 높은 추정 시간을 알아내는 것은 수사의 첫걸음이며 이는 디지털포렌식 수사에서도 마찬가지이다. PC에서 가장 많이 접하는 운영체제인 Windows에는 다양한 데이터를 타임스탬프와 함께 남긴다. 이러한 데이터를 시간순으로 나열, 사건 당시의 상황을 재현해 디지털 증거를 찾는 타임라인 분석 방법은 일반적인 디지털포렌식 분석 방법의 하나이다[1]. 사건 용의자는 타임스탬프를 수정하여 범행의 흔적을 숨긴다. 타임스탬프를 변경하는 도구들은 용의자들이 타임스탬프를 쉽게 조작할 수 있게 하며, 분석관들이 사건의 실제 순서를 재현하기 어렵게 만든다. 이를 대처하기 위하여 디지털포렌식 관점에서 수정된 타임스탬프를 탐지할 수 있는 여러 기법이 연구되고 있다. 본 논문은 다음의 두 가지 점에 착안하여 타임스탬프의 변조를 탐지하는 기법 설계를 제안하였다. 첫째로, 사건의 용의자는 파일의 타임스탬프를 의도적으로 변경하였을 경우, 1초 미만의 자릿값까지는 명확히 인지하지 못한다. 둘째로, 특정 시간 동안 저장장치에 기록된 데이터의 총용량은 그 저장장치의 최대 성능을 능가하지 못한다. 이를 위해 Windows에서 획득한 타임스탬프와 파일 크기 정보를 해당 기록이 저장된 저장장치의 읽기, 쓰기의 최댓값과 비교하여 시간의 변조를 탐지하고자 한다.

2. 배경 및 관련 연구

2.1 타임스탬프

New Technology File System(NTFS)은 기존의 File Allocation Table(FAT)을 대체하기 위해 1993년 Windows NT 3.1과 함께 발표되었으며 현재 Microsoft Windows의 보편적인 파일시스템이 되었다. NTFS에서 볼륨 내의 모든 파일과 디렉터리에는 적어도 하나 이상의 Master File Table(MFT)을 가지고 있는데 각 MFT 항목은 헤더와 해당 속성들과 관련된 속성 헤더를 포함한다. NTFS의 여러 표준 MFT 속성 중 타임스탬프는 $STANDARD_INFORMATION(SIA)과 $FILE_NAME(FNA)에 기록되어 있다. 이러한 타임스탬프는 용의자의 행위 추적에 사용되기 때문에 포렌식 분석에 많이 활용된다. SIA 및 FNA에는 각각 네 가지 고유한 타임스탬프가 있는데, 이 값들을 약어로 MACE(Modified, Accessed, Created, Entry Modified)로 표현하며 각각의 시간정보는 아래와 같다.

- Modified time(M): 파일 내용이 마지막으로 수정된 시간
- Accessed time(A): 파일을 마지막으로 접근한 시간
- Created time(C): 파일이 생성된 시간
- MFT Entry time(E): MFT 레코드가 마지막으로 업데이트된 시간

이후 내용에선 $FILE_NAME의 Modified time 표기 시 FNA-M과 같이 약어로 표현하기로 한다. SIA 및 FNA 타임스탬프를 포함한 모든 NTFS의 타임스탬프는 64비트 값으로 1601년 1월 1일 12:00 A.M. (UTC)을 시작으로 100나노초 단위의 정밀도[1]로 측정된다.

2.2 관련 연구

NTFS에서 타임스탬프 변조 탐지에 관한 기존의 연구는 크게 세 가지 범주로 요약된다. 첫째로 다양한 종류의 파일 또는 폴더에 기본 연산을 수행하고 패턴을 정리한 후 타임스탬프 규칙(R)을 만들어 변조를 탐지한다. 둘째로 시스템 아티팩트(A)에서 시간 변경이나 변조 도구 사용 시 생성되는 로그의 타임스탬프를 활용한다. 마지막으로 타임스탬프 변조 도구(T) 사용 시 생성되는 타임스탬프의 특이 패턴을 이용한다. 초기의 연구에서는 주로 기본 연산 패턴을 정리하고 Windows 버전별, 특정 파일 타입별 패턴의 특이점을 추가로 확인한 후 타임스탬프 규칙을 만들어 변조 여부를 종합적으로 판단하는 연구들이 수행되었다. 아티팩트는 $MFT와 $Logfile이 주로 사용되었다[2-7]. 최근의 연구에서는 기본 연산에서 파일 터널링이나 SIA-A 등의 활성화 여부에 따라 변화되는 타임스탬프 변화의 연구가 수행되었다[8]. 아티팩트는 $Usnjrnl, 프리패치, 링크 파일, 이벤트 로그, Volume Shadow Copies(VSC) 등을 활용한 연구로 확장되었다[9-10]. 타임스탬프 변조 도구 사용 여부를 탐지하는 연구도 진행되었다. 변조 도구들의 특성을 파악하여 각각의 유형별로 타임스탬프를 변조시켰을 경우 변화되는 특성과 한계점을 파악하여 시간의 변조 여부를 확인한다. 특정 변조 도구의 경우 타임스탬프의 초 미만의 자릿값이 모두 0으로 변경되는 특성을 활용하여 변조를 탐지한다[11-12]. Table 1은 기존 연구 내용을 요약하고 연구

1) 예로 64bit 헥사 값 "0x01DA0B237C55856D"는 "2023-10-30 20:23:19.8591341"이다

〈Table 1〉 Previous Researches

Researcher (Year)	Contents	R	A	T
Bang et al. (2011)	Timestamp rules in files and folders with 6 different Windows versions (Windows 2000, Server 2003, XP, Vista, Server 2008, 7)	O	-	-
Cho (2013)	10 timestamp patterns and 7 rules for detecting timestamp forgery in "txt", "docx" and "pdf" file types with analysis of $Logfile	O	O	-
Jang et al. (2016)	14 timestamp changing patterns and 7 rules for detecting timestamp modification with $MFT, $Logfile and manipulation tools	O	O	O
Palmbach and Breitinger (2020)	Reliabilities for detecting timestamp forgery of 5 artifacts($Logfile, Prefetch files, $Usnjrnl, Link files, Windows event logs)	-	O	-
Galhuber and Luh (2021)	Updating timestamp patterns with application-specific deviations and analyzing the effect and efficacy of timestamp forgery tools	O	O	O
Bouma et al. (2023)	File operations and modifiers(file tunneling, last access updating, transfer from FAT/exFAT, directories) analysis for reconstructing file histories	O	-	-

〈 R: Timestamp Rules, A: Artifacts, T: Timestamp Manipulation Tools 〉

가 어떠한 범주에 속하는지를 정리한 것이다. 이 외에도 iOS 등 다른 운영체제에서 시간 변조를 확인하거나 모바일이나 멀티미디어 기기에서 시간이 변조되는 행위를 파악하는 연구, 머신러닝을 활용하여 변조 여부를 확인하는 연구들도 수행되었다[13-15].

2.3 기존 연구의 한계점 및 본 연구의 착안점

용의자가 타임스탬프의 패턴을 정확하게 인지하고 정교하게 타임스탬프를 조작하면 탐지가 어려울 수 있다. 사건에서 핵심이 되는 파일의 생성시간을 인지하고 그 시간을 회피하여 시스템의 업데이트 시간과 같이 수많은 파일이 쓰인 시간대로 변경하면 변조 파일을 특정하기 쉽지 않다. 또한 아티팩트를 활용하여 타임스탬프의 변조를 탐지하는 연구의 경우엔 아티팩트가 용의자에 의해 쉽게 변경될 수 있으며 일정 기간이 지나면 로그는 시스템에 의해 자동으로 삭제되는 문제가 있다. 본 연구의 경우 초 미만의 타임스탬프 비트값을 사용, 스테가노그래피 채널로 활용하는 연구에서 설정한 가정과 같이, Windows의 파일 탐색기에서는 타임스탬프 정보를 초 단위까지만 볼 수 있어 초 미만의 자릿값은 사용자가 인식하지 못한다는 사실을 이용한다[16-17]. 또한, 사용자가 복사 연산을 수행하더라도 파일 탐색기를 통해선 복사의 총 쓰기시간을 인지하기 어렵다. 이러한 사실들을 바탕으로 한다면 용의자는 타임스탬프 변조 시 초 이하의 자릿값을 명확히 인지하지 못하고 저장장치의 읽기, 쓰기 성능을 고려하지 않은 채로 타임스탬프를 변경할 것이다. 따라서 본 연구는 용의자가 저장장치의 읽기, 쓰기 성능과 비교하여 생성될 수 없는 성능 이상의 타임스탬프 값으로 변경한 해당 타임스탬프를 탐지하여 변조를 확인하는 기법이다. 이는 기존 연구에서는 시도 되지 않은 방식이다.

3. 타임스탬프 변조 탐지 기법의 설계

3.1 설계 개요

본 연구에서는 저장장치의 성능을 활용하여 타임스탬프의 변조를 탐지하는 기법을 설계하였다. 저장장치마다 최대 읽기, 쓰기 속도를 가지고 있다는 점에서 실제로 Windows가 설치된 저장장치에서 파일이 초당 쓰일 수 있는 최댓값은 정해져 있다. 타임스탬프의 변조가 의심스러운 파일을 발견한 경우, 해당 파일이 쓰인 직후의 변경된 타임스탬프 시간 차이와 파일 크기를 통해 그 파일이 초당 기록된 데이터양을 확인할 수 있다. 그렇다면 이를 저장장치가 가진 최대 읽기, 쓰기 속도와 비교한다면 이론상 저장장치의 최대 속도를 초과할 수 없으므로 초과하면 그 파일의 타임스탬프는 변조된 것으로 본다.

Table 2는 1024MFile.txt 파일이 복사된 후 기록된 MFT 엔트리의 타임스탬프 헥사 값과 이를 시간 값으로 변환한 것이다. 파일의 복사 연산이 시작되면 복사 시작 시간은 SIA-C와 FNA-C, M, E, A 총 5개의 타임스탬프가 생성 시간인 2023-12-18 06:18:59.2287054로 변경된다. SIA-E 타임스탬프는 복사 완료 시간인 2023-12-18 06:19:00.6040953로 변경된다. SIA-M의 경우 이전 파일의 시간인 2023-11-12 02:11:12.0000000을 유지하며, SIA-A의 경우 복사 완료 1시간 내에서 시스템이

갱신 시간을 결정하는데 2023-12-18 06:19:15.0528196으로 갱신되었다. SIA-C와 SIA-E 타임스탬프의 시간차는 1.3753899초로 이 시간 동안 1024MB의 파일 크기를 가진 1024MFile.txt 파일은 평균 744.52MB/s의 속도로 쓰였다. 이 파일은 외장 SSD 드라이브에 저장된 파일을 C 드라이브의 SDD로 복사한 것으로, 외장 SSD의 최대 읽기속도는 1,000MB/s이며 C 드라이브의 SDD 최대 쓰기 속도는 3,300MB/s이다. 따라서 이 파일의 타임스탬프는 변조되지 않은 것으로 판단한다.

〈Table 2〉 Screen Shot of MFT Entry

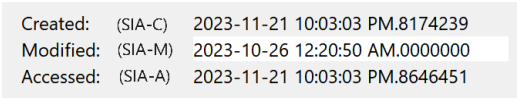
Hex Value

```
46 49 4C 45 30 00 03 00 F5 FB 77 D9 07 00 00 00  FILE0...õûwÙ....
5C 00 02 00 38 00 01 00 E0 01 00 00 00 04 00 00  \...8...à.......
00 00 00 00 00 00 00 00 04 00 00 00 D0 AA 00 00  ............Đª..
03 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00  ............`...
00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00  ........H.......
4E 31 64 16 7A 31 DA 01 00 20 F6 81 0D 15 DA 01  N1d.z1Ú.. ö...Ú.
79 0F 36 17 7A 31 DA 01 C4 C2 D2 1F 7A 31 DA 01  y.6.z1Ú.ÄÂÒ.z1Ú.
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00                   
00 00 00 00 86 0E 00 00 00 00 00 00 00 00 00 00  ....†...........
50 B3 05 12 01 00 00 00 30 00 00 00 78 00 00 00  P³......0...x...
00 00 00 00 00 00 03 00 5A 00 00 00 18 00 01 00  ........Z.......
E0 9D 01 00 00 00 02 00 4E 31 64 16 7A 31 DA 01  à.......N1d.z1Ú.
4E 31 64 16 7A 31 DA 01 4E 31 64 16 7A 31 DA 01  N1d.z1Ú.N1d.z1Ú.
4E 31 64 16 7A 31 DA 01 00 00 00 00 40 00 00 00  N1d.z1Ú....@....
00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00  ........ .......
0C 02 31 00 30 00 32 00 34 00 4D 00 46 00 7E 00  ..1.0.2.4.M.F.~.
31 00 2E 00 54 00 58 00 54 00 00 00 00 00 00 00  1...T.X.T.......
```

Explanation

SIA	C	2023-12-18 06:18:59.2287054	M	2023-11-12 02:11:12.0000000
	E	2023-12-18 06:19:00.6040953	A	2023-12-18 06:19:15.0528196
FNA	C	2023-12-18 06:18:59.2287054	M	2023-12-18 06:18:59.2287054
	E	2023-12-18 06:18:59.2287054	A	2023-12-18 06:18:59.2287054

Fig. 1은 Windows 파일 탐색기에서 확인할 수 있는 타임스탬프 값이다. 용의자가 타임스탬프의 변조를 시도한다고 하였을 때 파일 탐색기는 SIA-C, M, A값만을 보여주기 때문에 SIA-E 타임스탬프는 용의자가 인지하지 못한다. 타임스탬프의 시간 차가 많이 나지 않는 파일을 변조할 경우, 타임스탬프의 시간 차이가 크게 나지 않기 때문에 초 미만의 자릿값을 저장장치의 최대 성능까지 고려하여 정교하게 변조하지는 못하고 무작위로 변경할 가능성이 크다. 그렇다면 그러한 타임스탬프는 본 기법에 탐지될 가능성이 있다.



[Fig. 1] Windows File Explorer Timestamp

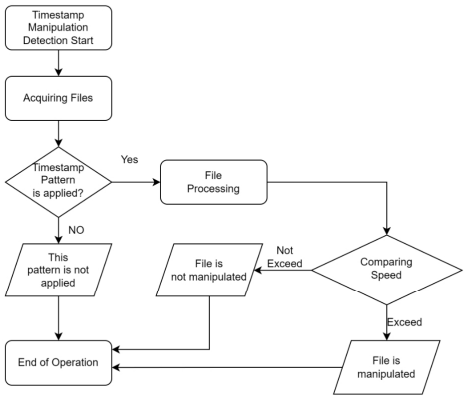
3.2 탐지 알고리즘

Table 3은 탐지 알고리즘을 표현한 간단한 의사 코드이다. 1번째 칼럼에서 분석하고자 하는 파일의 경로를 확인 후 파일을 획득한다. 2번째 칼럼에서는 획득된 파일이 타임스탬프 패턴 분석을 통한 변조 탐지가 가능한지 판단하고 아니라면 타임스탬프 패턴을 분석할 수 없다는 결과를 출력하고, 그러하다면 3-5의 칼럼에서와 같이 SIA-C와 SIA-E 시간 그리고 파일의 크기를 가져오는 함수를 호출하여 각각의 변수에 저장한다. 6번째 칼럼에서 SIA-E과 SIA-C의 시간 차이를 계산하여 쓰기 시간 변수에 저장한다. 이는 파일에 대한 쓰기 작업이 발생한 시간이다. 7번째 칼럼에서는 파일 크기의 값과 쓰기시간 값을 나누기 연산을 수행하고 그 결괏값을 파일 쓰기 속도 변수에 저장한다. 8번째 칼럼에서는 최대 읽기, 쓰기 속도를 나타내는 값을 저장장치와 최대 속도를 데이터베이스 파일에서 가져온다. 9번째 칼럼에서 파일 쓰기 속도가 최대 읽기, 쓰기 속도보다 크다면 이후 칼럼에서 파일은 조작된 것으로 판단하고 결과를 출력하며, 아니라면 조작되지 않는 파일로 판단하고 결과를 출력한다.

〈Table 3〉 Pseudo Code

```
1   file_path = "path/to/file"
2   if timestamp_pattern_applied(file_path) {
3     c_time = getCreatedTime(file_path)
4     e_time = getMFTEntryTime(file_path)
5     file_size = getFileSize(file_path)
6     write_time = e_time − c_time
7     file_write_speed = file_size / write_time
8     max_rw_speed = getFromDatabase("max_rw_speed")
9     if (file_write_speed 〉 max_rw_speed) {
10          output "File is manipulated."
11    } else {
12          output "File is not manipulated."
13          }
14  else
15     output "This pattern is not applied."
16  }
```

Fig. 2는 탐지 알고리즘의 의사 코드를 순서도로 표기한 것이다. 파일 획득 후에 타임스탬프 패턴을 살펴 변조 탐지의 가능 여부를 판단한 후 파일 탐지에 사용되는 변수들을 연산 처리하고 속도를 비교하는 전체 프로세스는 같고, 변수를 불러온 후 원하는 값을 계산하는 항목들은 File Processing 항목으로 단일화하고 속도를 비교하는 연산을 Comparing Speed로 단순화하였다.



[Fig. 2] Flowchart

3.3 본 기법의 한계점

본 기법은 쓰기 작업이 발생하는 파일 복사, 다른 볼륨에서 파일의 이동과 같은 일부 연산에만 적용되므로 모든 연산 작업에 활용할 수 없다. 따라서 이 기법은 기존의 타임스탬프 패턴의 원칙을 우선 적용하여 파일스탬프 변조 여부를 판별하고 세부적으로 파일 변조가 의심스러운 쓰기 작업이 발생된 파일에 한해 부가적으로 활용하여 타임스탬프 변조 탐지의 정확성을 높이는 데 사용되어야 할 것이다. 또한 특정 크기 이하들의 파일들이 저장장치에 쓰일 경우, 저장장치는 속도 향상을 위해 캐싱 기법을 사용하기 때문에 파일이 한순간에 쓰이게 되고 그 결과 미세한 쓰기의 시간 차를 타임스탬프에 반영하지 못하고 SIA-C와 SIA-E의 타임스탬프값이 동일하게 생성되는 결과를 가져온다. 이의 경우 본 기법의 적용이 불가하다.

4. 결론

본 논문은 기존의 타임스탬프 변조 기법에서 사용되지 않았던 저장장치의 최대 읽기, 쓰기 속도를 활용하여 타임스탬프의 변조를 확인하는 기법을 설계하였다. 이 기법 설계는 아티팩트의 활용이 불가하여 타임스탬프 패턴만으로 변조의 여부를 판단하고자 할 때 복사 등 쓰기 연산 패턴에서 파일이 쓰인 총시간과 파일 크기를 계산하고 이를 저장장치의 속도와 비교하여 최댓값을 초과하는 경우를 타임스탬프 변조로 판단함으로써 기존 타임스탬프 변조 탐지에 더하여 정확성을 높이는 데 활용된다. 향후 연구로는 이 기법 설계를 반영한 실험에서 저장장치와 테스트 파일의 종류 그리고 파일의 용량을 다양화하여 쓰기 이벤트를 발생시켜, 본 연구가 적용될 수 있는 최소단위의 파일 단위를 특정하는 연구를 진행하고자 한다.

REFERENCES

[1] K.G.Lee, S.J.Hwang, C.H.Lee and S.J.Lee, "Study on advanced analysis method based on timeline chart for Digital Forensic Investigation," Journal of Advanced Navigation Technology, Vol.18, pp.50-55, 2014.

[2] J.W.Bang, B.Y.Yoo and S.J.Lee, "Analysis of changes in file time attributes with file manipulation," Digital Investigation, Vol.7, No.3, pp.135-144, 2011.

[3] G.S.Cho, "A computer forensic method for detecting timestamp forgery in NTFS," Computers & Security, Vol.34, pp.36-46, 2013.

[4] G.S.Cho, "A Digital Forensic Method by an Evaluation Function Based on Timestamp Changing Patterns," Journal of the Korea Society of Digital Industry and Information Management, Vol.10, No.2, pp.91-105, 2014.

[5] D.I.Jang, G.J.Ahn, H.U.Hwang and K.B.Kim, "Understanding Anti-forensic Techniques with Timestamp Manipulation," 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI), pp.609-614, 2016.

[6] H.J.Yoon, "A study on user behavior tracking using $UsnJrnl," Master's Thesis, Seoul University, 2018.

[7] J.H.Jeong, "A study on document reading and writing traces through NTFS file system journal file analysis," Master's Thesis, Seoul University, 2022.

[8] J.Bouma, H.Jonker, V.Meer and E.Aker, "Reconstructing Timelines: From NTFS Timestamps to File Histories," ARES '23: Proceedings of the 18th International Conference on Availability, Reliability and Security, No.154, pp.1-9, 2023.

[9] D.Palmbach and F.Breitinger, "Artifacts for Detecting Timestamp Manipulation in NTFS on Windows and Their Reliability," Forensic Science International: Digital Investigation, Vol.32, Supplement. pp.S1-S9, 2020.

[10] A.Mohamed and C.Khalid, "Detection of Suspicious Timestamps in NTFS using Volume Shadow Copies," International Journal of Computer Network and Information Security, Vol.13, Issue4, pp.62-69, 2021.

[11] M.Galhuber and R.Luh, "Time for Truth: Forensic Analysis of NTFS Timestamps," ARES '21: Proceedings of the 16th International Conference on Availability, Reliability and Security, No.44, pp.1-10, 2021.

[12] G.S.Cho, "A Digital Forensic Analysis of Timestamp Change Tools for Windows NTFS," Journal of The Korea Society of Computer and Information, Vol.24, No.9, pp.51-58, 2019.

[13] S.H.Lee, Y.H.Lee and S.J.Lee, "A study on the Evidence Investigation of Forged/Modulated Time-Stamp at iOS(iPhone, iPad)," KIPS Transactions on Computer and Communication Systems, Vol.5, No.7, pp.173-180, 2016.

[14] J.H.Han and S.J.Lee, "A Study on the Processing of Timestamps in the Creation of Multimedia Files on Mobile Devices," Journal of Information Processing Systems, Vol.18, No.3, pp.402-410, 2022.

[15] A.Mohamed and C.Khalid, "Detection of Timestamps Tampering in NTFS using Machine Learning," Procedia Computer Science, Vol.160, pp.778-784, 2019.

[16] S.Neuner, A.G.Voyiatzis, M.Schmiedecker, S.Brunthaler, S.Katzenbeisser and E.R.Weippl, "Time is on my side: Steganography in filesystem metadata," Digital Investigation, Vol.18, Supplement7, pp.76-86, 2016.

[17] G.S.Cho, "Data Hiding in NTFS Timestamps for Anti-Forensics," International Journal of Internet, Broadcasting and Communication, Vol.8, No.3, pp.31-40, 2016.

송 종 화(Jong-Hwa Song) [준회원]

■ 2004년 8월 : 백석대학교 정보통신학부 컴퓨터학과 (공학사)
■ 2018년 3월 ~ 현재 : 성균관대학교 과학수사학과 대학원생

〈관심분야〉
과학수사, 파일시스템, 디지털포렌식

이 현 섭(Hyun-Seob Lee) [종신회원]

■ 2013년 2월 : 한양대학교 컴퓨터공학과 (공학 박사)
■ 2012년 3월 ~ 2021년 2월 : 삼성전자 책임연구원
■ 2021년 3월 ~ 현재 : 백석대학교 컴퓨터공학부 조교수

〈관심분야〉
인공지능, 저장시스템, 임베디드 시스템