The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 16, No. 3, pp.21-28, Jun. 30, 2016. pISSN 2289-0238, eISSN 2289-0246

http://dx.doi.org/10.7236/JIIBC.2016.16.3.21
JIIBC 2016-3-4

리눅스 시스템에서의 파일 조작에 따른 시간변화 분석

Analysis of File Time Change by File Manipulation of Linux System

유병영*
Byeongyeong Yoo*

요 약 디지털 포렌식 조사에서 파일의 시간정보는 중요한 의미를 가진다. 리눅스의 Ext4(Extended File System 4) 환경에서 획득할 수 있는 파일의 시간정보는 파일 접근 시간(Access Time), 수정 시간(Modification Time), Inode 변경 시간(Change Time), 삭제 시간(Deletion Time), 생성 시간(Creation Time)이다. 일반적으로 파일의 생성, 수정, 복사 등 여러 가지 행위에 따라 시간 정보가 변화되며, 증거 분석을 위해 행위에 따른 파일 시간변화에 대한 연구가 필요하다. 본 논문에서는 리눅스 Ext4 환경에서 파일 및 폴더의 다양한 행위에 따른 시간정보를 분석하였고 이를 이용하여 악성코드의 실제 감염시간과 파일의 변조 여부를 확인하는 방안을 연구하였다.

Abstract File Time information has a significant meaning in digital forensic investigation. File time information in Linux Ext4 (Extended File System 4) environment is the Access Time, Modification Time, Inode Change Time, Deletion Time and Creation Time. File time is variously changed by user manipulations such as creation, copy and edit. And, the study of file time change is necessary for evidence analysis. This study analyzes the change in time information of files or folders resulting from user manipulations in Linux operating system and analyzes ways to determine real time of malware infection and whether the file was modulation.

Key Words : Digital Forensic, File System, File Time

Ⅰ. 서 론

디지털 포렌식 조사에서 파일 시간 정보는 중요한 의미를 가진다. 일반적으로 파일의 생성, 수정, 복사 등 여러 가지 행위에 따라 시간 정보가 변화되며, 파일의 시간 분석을 통해 사용자의 행위를 유추할 수 있고, 디지털 포렌식 수사시 중요한 증거로 사용될 수 있다.

파일의 시간정보는 운영체제와 파일시스템의 종류 및 버전에 따라 다른 정보를 가진다. 윈도우 운영체제 NTFS(New Technology File System)는 일반적으로 사용자가 확인할 수 있는 $Standard_Information 속성의 파일 생성, 수정, 접근 시간 외에, Entry 수정 시간, $File_Name 속성의 파일 생성, 수정, 접근, Entry 수정 시간에 대한 정보를 저장하고 있다[1]. 이러한 추가 시간 정보는 사용자 행위 분석 등 디지털 포렌식 조사에서 중요한 증거로 활용된다. 따라서 리눅스 운영체제의 Ext4 환경에서도 파일 속성에서 일반적으로 확인 할 수 있는 시간 외에 추가적인 시간정보에 대한 확인이 필요하다.

윈도우 운영체제에서 주로 사용되는 NTFS, FAT 파일시스템에 대해서는 행위에 따른 파일 및 폴더의 시간

*정회원, 국방과학연구소 연구원
접수일자: 2016년 5월 2일, 수정완료일: 2016년 6월 2일
게재확정일자 : 2016년 6월 10일

Received: 2 May, 2016 / Revised: 2 June, 2016
Accepted: 10 June, 2016
*Corresponding Author: forensic@add.re.kr
Dept. of Computer Science, Agency for Defense Development, Korea

변화를 분석하고 이를 활용하는 방안이 이미 연구되었다[1][2]. 이를 이용하여 실제 조사에서 증거 분석에 다양하게 활용 되고 있다. 리눅스 운영체제는 윈도우에 비해 점유율은 떨어지지만 서버 부분에서 높은 비중으로 사용 중이며, 디지털 포렌식 조사의 분석 대상으로서 중요한 가치를 가진다. 이에 본 논문에서는 리눅스 운영체제 Ext4 환경에서 행위에 따른 파일 및 폴더 시간정보를 분석하고 이를 증거 분석에서 활용할 수 있는 방안을 제시한다.

본 논문은 다음과 같이 구성된다. 2절에서는 본 논문과 관련된 기존의 연구들에 대해서 설명하고, 3절에서는 Ext4의 내부 구조와 시간정보를 분석한다. 4절에서는 다양한 행위에 따른 파일 및 폴더의 시간 변화를 분석한다. 5절에서는 4장에서 획득한 시간 변화 정보를 이용한 침해사고 및 디지털 포렌식 조사 방안을 소개한다. 그리고 6절에서 결론을 내리고 추가 연구 방안을 제시한다.

II. 관련 연구

디지털 포렌신 관점에서 Ext4에 대한 분석은 주로 삭제된 파일을 복구하는 방안이 연구되었다. Ext2 파일 시스템은 NTFS, FAT 파일 시스템과 같이, 파일 삭제 시 메타데이터의 정보가 남아있게 된다. 남아있는 삭제된 파일의 정보를 이용하여 파일을 복구하는 연구가 진행되었고[3][4], 메타데이터를 이용하여 파일을 복구할 수 있는 것이 확인 되었다. 그러나 Ext3부터는 파일 삭제시 해당 메타데이터를 초기화하기 때문에 기존 방법으로는 복구가 어려웠다.

Ext3 파일시스템에서 삭제된 파일 복구에 관한연구는 주로 Ext3 파일시스템이 사용한 파일 저장방식인 inode table에 존재하는 indirect block pointer의 정보를 통해 비할당 영역에서 조각난 파일들의 offset을 추정하여 복구하는 연구가 진행되었고[5][6][7], ext3grep, extcarve와 같은 대부분의 Ext3 파일시스템복구 도구는 이러한 방식을 사용한다. 하지만Ext4 파일시스템에서는 Ext3와는 달리 파일의 데이터블록포인터 정보를 extents 구조체를 사용하여 저장하기 때문에 Ext3 파일시스템에서 사용한 파일복구 방법을 적용할 수 없다.

Ext4 파일시스템에서 삭제된 파일의 복구에 대한 연구로 삭제된 파일의 inode table에 남아있는 extents index node의 offset을 획득하여 삭제된 파일을 복구하는 연구가 진행 되었지만 파일의 extents tree의 깊이가 0일 경우에는 extents index node가 존재하지 않고 extents tree의 깊이가 1 이상일 경우 extents index node의 offset을 획득하더라도 실제 블록들의 offset 정보는 초기화되기 때문에 삭제된 파일의 데이터 블록을 확인하기 어렵다[8].

추가적으로 Ext3부터 존재하는 journal log 영역에 남아 있는 삭제 트랜잭션에 존재하는 Block Bitmap 정보를 이용하여 삭제된 블록을 복구하는 방법도 진행되었다[9]. 그리고 안드로이드 운영체제에서 사용하는 Ext4에서 안드로이드 내부 사용자 데이터가 존재하는 파티션, 애플리케이션 설치 시 생성되는 파일들의 종류 및 특성 등을 이용하여 삭제된 관련 파일을 복구하는 방법이 연구 되었다[10].

이처럼 디지털 포렌식 관점의 Ext4 연구는 삭제된 파일의 복구 방안에 대한 연구가 주를 이루었다. Ext4의 파일 시간과 관련된 연구는 Inode Table의 포맷을 분석하여 저장된 시간 정보를 확인한 연구가 있었지만[8], 사용자 및 시스템의 행위 분석을 위한 파일의 시간 변화 연구는 진행 되지 않았다. 이와는 달리 Windows 운영체제 대상으로는 파일 및 폴더의 다양한 조작에 따른 시간 변화가 연구되었다[1][2]. 해당 연구는 Windows 운영체제의 NTFS, FAT 파일 시스템을 대상으로 시간 변화를 분석하였고, 이를 이용하여 파일 시간조작 여부 및 사용자 행위를 확인할 수 있는 방안을 제시하였다. 마찬가지로 리눅스 운영체제의 Ext4 환경에서도 파일 및 폴더의 조작에 따른 시간 변화 분석이 필요하며 이를 활용한 디지털 포렌식 조사 방안의 연구가 필요하다.

III. Ext4 - 리눅스 파일 시스템

Ext4는 Extended File System 4의 약자로 리눅스 시스템에서 사용되는 파일시스템이다. 1992년 4월 Ext1로 처음 적용되었으며 Ext2, Ext3를 거쳐 지금의 Ext4가 널리 사용되고 있다. Ext4는 64비트 기억 공간 제한을 없애고, Ext3의 성능을 향상시키며, 하위 호환성이 있는 확장 버전으로 개발되었다[8]. 이는 Ext2, Ext3에서 쉽게 업그레이드 가능하도록 설계 되었으며, 파일 조각화 현상을 최소화 하였다. 또한 안드로이드 OS 2.3(Gingerbread)부터 공식 파일시스템으로 지정되어 사용되어 지고 있다.

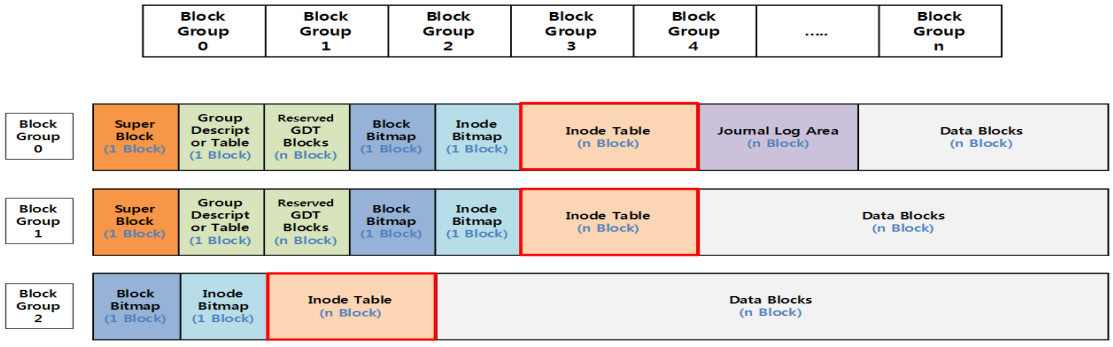

그림 1. Ext4 파일시스템 구조
Fig. 1. Ext4 file system structure

1. Ext4 구조

파일 시스템은 파일 데이터 내용뿐만 아니라 파일의 이름, 시간, 크기 및 저장매체 상의 위치와 같은 데이터도 함께 생성하여 해당 파일 관리를 위해 사용한다. 이러한 데이터를 메타 데이터(Meta data)라 하며 이를 생성하고 관리하는 방법은 파일 시스템마다 다르다.

Ext4는 데이터를 저장하는 기본단위로 블록(Block)을 사용한다. 블록은 NTFS, FAT 파일시스템의 클러스터와 유사하다고 할 수 있으며, 블록의 크기는 1KB, 2KB, 4KB로 사용 될 수 있으며, 기본적으로 4KB를 사용한다. Ext4는 여러 블록들을 블록 그룹(Block Group)으로 묶어서 관리한다. 모든 블록 그룹들의 크기는 동일하며 블록 그룹의 개수는 디스크 크기에 따라 다르다[10].

하나의 블록 그룹은 최대 32,768개의 블록을 관리하기 때문에 블록의 크기가 4KB일 경우 하나의 블록 그룹의 크기는 최대 128MB이다. 만약 Ext4 파일 시스템의 파티션의 크기가 1GB(1,048,576MB)일 경우 8개의 블록 그룹(128MB × 8 = 1,048,576MB)이 존재하게 된다.

그림 1은 Ext4 파일시스템의 구조를 나타낸다. 블록 그룹 3부터는 블록 그룹1과 2의 형태가 번갈아가며 반복되는 구조를 가진다[10].

블록 그룹 내부에는 Super Block, Group Descript or Table, Block Bitmap, Inode Bitmap, Inode Bable, Journal Log Area 등의 메타데이터 영역들이 존재한다. Super Block은 파일시스템의 전체적인 정보를 저장하는 블록으로 블록의 크기, 전체 블록의 개수, Inode Table의 크기, 블록 그룹 당 Inode Table의 개수 등을 저장하고 있다. Group Descript or Table은 각 블록 그룹에 대한 Block Bitmap, Inode Table의 시작 블록 번호, 사용하지 않는 블록 정보 등을 저장한다. Block Bitmap은 블록 그룹안의 블록들의 할당 현황을 저장하며, 하나의 블록을 하나의 비트로 표현한다. Inode Bitmap은 블록 그룹안의 Inode 번호의 할당 현황을 저장하며, 하나의 Inode 번호를 하나의 비트로 표현한다. 각 블록 그룹 내 Inode

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Mode		Uid		파일 크기				파일 접근 시간(Access Time)				Inode 변경 시간(Change Time)			
0x10	파일 수정 시간(Modification Time)				파일 삭제 시간(Deletion Time)				Gid		하드 링크 수		데이터를 저장하는데 필요한 블록 개수			
0x20	Flags				[union] Version, translator, reserved → OS dependent											
0x30																
0x40																
0x50																
0x60					File version (for NFS)				File ACL				S_size_high			
0x70	Obsoleted fragment address				Union											
0x80	Extra_isize		Pad1		Extra Change Time				Extra Modification time				Extra Access time			
0x90	파일 생성 시간(Creation time)				Extra File Creation time											

그림 2. Inode Table 블록 구조
Fig. 2. Inode Table block structure

Table 블록은 하나의 파일에 해당하는 메타데이터를 저장하며, 파일의 시간, 크기, 데이터 블록 개수 등의 정보를 저장한다. Journal Log Area는 전체 블록 그룹 중 하나의 블록 그룹에만 존재하며, 파일시스템에서 발생한 트랜잭션으로 생성된 저널 로그들이 존재한다. Data Blocks는 여러 개의 블록으로 구성되며 Inode 번호, 파일 타입, 파일 이름 등의 메타데이터를 저장하는 Directory Entry와 실제 데이터를 저장하고 있다. Inode Table에 존재하는 메타데이터와 결합하여 하나의 파일에 대한 완전한 메타데이터를 형성한다[10].

2. Ext4 파일 시간정보

그림 2는 Inode Table의 구조를 나타낸다. Inode Table에는 파일 접근 시간(Access Time), Inode 변경 시간(Change Time), 수정 시간(Modification Time), 삭제 시간(Deletion Time), 생성 시간(Creation Time)의 5가지 시간정보를 저장된다[8].

파일 접근 시간, Inode 변경 시간, 수정 시간은 일반적으로 MACTime으로 불리며, 파일의 여러 가지 변화에 따라 다양하게 시간이 변경된다. 해당 시간은 stat과 같은 리눅스 기본 명령어로 확인이 가능하며, utimes, futimes 등의 명령어로 변경이 가능하다. 파일 삭제 시간은 파일 삭제시의 시간이 저장되며, 파일이 삭제되기 전까지는 데이터를 저장하지 않는다. 파일 생성 시간은 Ext4에서 추가된 시간 정보로 파일의 생성시의 시간이 저장되며 일반적으로 변경되지 않는다. 파일 생성 시간은 일반적으로 파일시스템의 지식 없이는 확인과 조작에 어려움이 따른다.

Inode Table에 저장되는 시간정보는 4bytes 크기로 저장된다. 4bytes 크기에 저장될 수 있는 시간 정보는 1970년도부터 2038년까지로 한정되어 있다. 이를 해결하기 위해 Extra Change Time, Extra Modification Time, Extra Access Time, Extra File Creation Time 데이터가 Ext4에 추가되었다. 각 시간정보는 4bytes의 크기를 가지며, 하위 2bits는 기존 4bytes 시간에 더해져서 시간 확장을 위해 사용된다. 나머지 30bits는 nanoseconds 단위의 시간정보를 저장한다.

Ⅳ. Ext4 환경의 파일 시간 변화 분석

리눅스 Ext4 환경에서 다양한 행위에 따른 시간 정보 변화를 확인하기 위해 표 1과 같은 환경에서 실험을 진행하였다. 운영체제는 리눅스 중 데비안 계열의 CentOS, 레드햇 계열의 Ubuntu 최신 버전을 대상으로 선정하고, 파일 시간 정보 확인을 위해 EnCase, Autopsy 도구를 활용하였다. 파일 시간 변화에 대한 정확한 확인을 위해 새로운 파일을 생성, 파일 조작 후에 디스크의 이미지를 획득한 후, EnCase, Autopsy를 이용하여 시간정보를 확인하였다.

표 1. 파일 시간 변화 분석 환경
Table 1. environment of file time change analysis

OS	◦ CentOS 7.2.1511 x86_64 ◦ Ubuntu 16.04 AMD64 desktop			
Tools	◦ Guidance Software EnCase v7.12 ◦ Brian Carrier's Autopsy v3.1.3			
Volume	/dev/hda1		/dev/hdb1	
	C:\	200GB	D:\	100GB

1. Ext4 파일 시간 변화

표 2는 행위에 따른 파일 시간 변화를 보여준다. O표시가 된 칸은 행위가 일어난 시간으로 파일의 정보가 변경되는 것을 의미하며, C는 CentOS, U는 Ubuntu 운영체제만 시간정보가 변경되는 것을 의미한다. 일반적으로 파일 접근, 수정, Inode 변경 시간은 행위에 따라 다양하게 변화되는 것을 확인할 수 있다. 반면 파일 생성 시간

표 2. 행위에 따른 파일 시간 변화
Table 2. File time change by behavior

행 위	접근	수정	Inode 변경	파일 생성
파일 생성	O	O	O	O
파일 이름 변경		O	U	
파일 확장자 변경		O	C	
파일 휴지통으로 이동		O		
파일 내용 수정 (vi)	O	O	O	
파일 내용 수정 (gedit)	O	O	O	C
파일 권한 수정		O	U	
파일 복사 - 동일 볼륨		O	O	O
파일 이동 - 동일 볼륨		O	C	
파일 복사(cp 명령어) - 동일 볼륨	O	O	O	O
파일 이동(mv 명령어) - 동일 볼륨		O	C	
파일 복사 - 다른 볼륨		O	O	O
파일 이동 - 다른 볼륨		O	O	O
파일 복사(cp 명령어) - 다른 볼륨	O	O	O	O
파일 이동(mv 명령어) - 다른 볼륨		O	O	O

은 파일 생성, 복사와 같이 파일이 새로 생성되는 경우에만 시간이 변화되는 것을 확인할 수 있다. 예외적으로 CentOS에서 gedit를 이용한 파일 내용 수정 행위에서 파일 생성 시간이 변화되는 것이 확인 되었다. 해당 행위에서 파일 생성 시간이 변화되는 이유는 gedit를 이용해 파일을 open할 경우 원본 파일은 파일의 이름을 변경하여 임시 백업 파일로 저장하고 새로 작업하는 내용은 원래 파일의 이름으로 새로 생성되기 때문에, 수정 후 파일 저장시 새로운 파일이 저장되는 것과 같은 시간 변화를 보인다.

확장자가 없는 파일의 이름 변경시 확장자가 변경으로 간주됨을 확인 하였고, 동일 및 다른 볼륨으로 파일 복사시 원본파일의 시간 정보 변화는 없는 것으로 확인되었다. 또한 같은 Ext4 환경이라도 OS의 종류에 따라 일정 부분에서 시간정보가 다르게 저장됨을 확인할 수 있다.

2. Ext4 폴더 시간 변화

표 3은 행위에 따른 폴더 시간 변화를 보여준다. 파일 시간 변화와 마찬가지로 파일 접근, 수정, Inode 변경 시간은 행위에 따라 다양하게 변화되는 것을 확인할 수 있으며, 파일 생성 시간도 폴더 생성, 복사와 같이 새로 생성되는 경우에만 시간이 변화되는 것을 확인 하였다. 그리고 폴더를 동일 또는 다른 볼륨으로 복사/이동시 폴더 내부의 파일과 폴더의 시간에 영향을 주는 것을 확인 하였다.

표 4는 폴더 복사/이동에 따른 내부 파일 및 폴더의 시간 변화를 나타낸다. 이중 다른 볼륨으로 폴더를 이동하는 경우의 내부 파일은 Inode 변경 시간이 수정되고, 내부 폴더는 파일 접근, Inode 변경 시간이 수정된다.

표 3. 행위에 따른 폴더 시간 변화
Table 3. Folder time change by behavior

행 위	접근	수정	Inode 변경	파일 생성
폴더 생성	O	O	O	O
폴더 이름 변경		O		
폴더 권한 수정		O		
폴더 복사 - 동일 볼륨		O	O	O
폴더 이동 - 동일 볼륨		O		
폴더 복사(cp 명령어) - 동일 볼륨	O	O	O	O
폴더 이동(mv 명령어) - 동일 볼륨		O	C	
폴더 복사 - 다른 볼륨		O	O	O
폴더 이동 - 다른 볼륨	C	O	O	O
폴더 복사(cp 명령어) - 다른 볼륨	O	O	O	O
폴더 이동(mv 명령어) - 다른 볼륨		O	O	O
파일 복사(이동)시 대상 부모 폴더	O	O	C	
파일 이동시 원본 부모 폴더 (파일 복사는 변화 없음)	O	O	O	

표 4. 폴더 복사/이동에 따른 내부 파일 시간 변화
Table 4. File time change in the folder by folder copy and move

행 위	접근	수정	Inode 변경	파일 생성
폴더 복사 - 동일 볼륨	O		O	
폴더 이동 - 동일 볼륨				
폴더 복사(cp 명령어) - 동일 볼륨	O	O	O	
폴더 이동(mv 명령어) - 동일 볼륨				
폴더 복사 - 다른 볼륨	O		O	
폴더 이동 - 다른 볼륨	O		O	
폴더 복사(cp 명령어) - 다른 볼륨	O	O	O	
폴더 이동(mv 명령어) - 다른 볼륨			O	

V. 시간 변화 정보를 활용한 조사 방안

1. 침해사고 조사 활용 방안

악성코드는 다양한 방법을 통해 감염여부를 확인하기 어렵게 하며[11], 감염 시간을 알기 어렵게 하기 위해 감염시 악성코드의 시간정보를 조작 하는 경우가 많다. 이러한 경우 악성코드의 감염 시간은 침해사고 조사에 중요한 요소로 작용한다.

Operation Windigo 공격은 유닉스 및 리눅스 서버를 해킹하여 악성코드를 감염시킨 후 SSH 계정 탈취, 악의적인 웹페이지 연결, 스팸메일 발송 등의 악성 행위를 수행하는 것으로 확인 되었다[12]. 그리고 악성코드 분석 결과 감염시 악성코드의 시간정보를 utimes 명령어를 통해 변경하는 것을 확인하였다. utimes 명령어는 파일 접근, 수정 시간 값을 인자로 입력받아 해당 시간으로 변경한다. 이러한 경우 Inode 변경 시간은 명령어가 수행된 시간으로 변경된다.

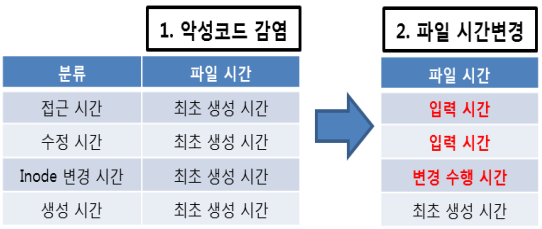

그림 3. 감염된 악성코드 시간 변화
Fig. 3. Infected malware time change

그림 3은 Operation Windigo 악성코드가 감염될 때 시간정보와 utimes 명령어를 이용하여 시간을 변경한 후의 악성코드 시간 정보를 나타낸다. 파일 시간 변경시 파일 접근, 수정 시간은 입력한 임의 시간으로 변경되고, Inode 변경 시간 명령어를 수행한 시간으로 변경 된다. 그러나 파일 생성 시간은 변하지 않고 최초 파일 생성 시간을 그대로 저장하고 있는 것을 확인할 수 있다. 이를 통해 악성코드의 실제 감염 시간을 확인할 수 있으며, 침해사고의 전체적인 시간 관계를 구성할 수 있다.

2. 디지털 포렌식 조사 활용 방안

디지털 포렌식 조사에서 파일의 조작 여부는 중요한 증거가 될 수 있다. 리눅스 시스템은 주로 개인 PC 보다는 서버에서 주로 사용된다. 이러한 경우 일반 문서파일이나 사용자 생성 파일보다는 시스템에서 생성한 이벤트 및 행위 로그가 중요한 정보로 작용되며 해당 파일의 조작여부를 확인하는 것은 조사에서 중요한 요소로 작용한다[13].

시스템 로그를 조작하기 위해서는 파일을 삭제하고 다른 로그를 복사하는 작업을 수행하거나, 파일을 열어서 직접 내용을 수정하는 작업을 거친 후, utimes, futimes 등의 시간 변경 명령어를 통해 파일 접근, 수정, Inode 변경 시간을 조작하는 과정을 거친다.

일반적으로 시스템 로그 파일은 어떤 행위에 대한 메시지와 해당 행위가 발생한 시간을 저장하고 있다. 정상 파일의 경우 로그 메시지의 발생 시간과 파일의 Inode에 존재하는 파일 생성 시간(Creation Time)이 동일해야 한다. 만약 파일 생성시간이 로그 메시지의 시간보다 늦은 시간일 경우 파일을 조작했다고 판단 할 수 있다.

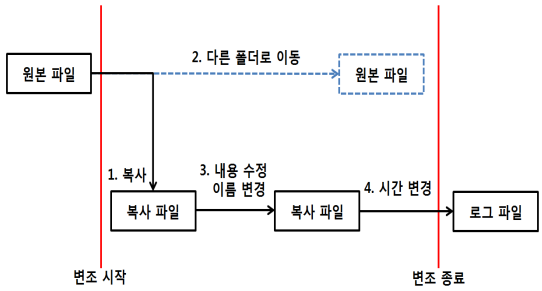

그림 4. 파일 변조 과정(파일 치환)
Fig. 4. File modulation process(File replacement)

그림 4는 복사한 원본 파일을 조작하는 과정을 보여준다. 사용자는 원본 파일을 복사한 후 원본 파일 보관을 위해 다른 폴더로 이동한다. 그 후 복사 파일을 대상으로 로그의 내용을 수정하고 파일의 이름을 원본 파일과 동일하게 변경한다. 마지막으로 utimes 등의 명령어를 이용하여 복사 파일의 시간을 변경한다. 이러한 경우 복사 파일의 생성 시간이 파일 복사 시점으로 설정되기 때문에 이를 로그 메시지 시작 시간과 비교하여 조작 여부를 판단 할 수 있다.

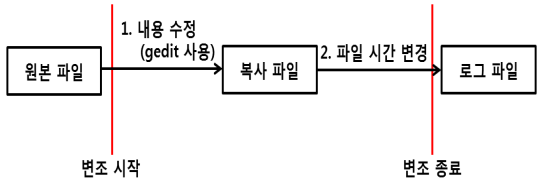

그림 5. 파일 변조 과정(원본 파일 수정)
Fig. 5. File modulation process(Original file modification)

로그 파일을 열어서 직접 내용을 수정하는 경우 vi, gedit를 사용하는 등 여러 가지 방법이 존재한다. 이중 모든 행위에 대해 로그 파일의 조작 여부를 판단하는 것은 불가능 하나, 그림 5와 같이 CentOS에서 gedit을 이용하여 로그 파일을 수정한 경우 조작여부 판단이 가능하다. CentOS에서 gedit을 이용해 파일을 수정할 경우 원본 파일은 파일의 이름을 변경하여 임시 백업 파일로 저장하고 새로 작업하는 내용은 원래 파일의 이름으로 새로 생성되기 때문에, 수정 후 파일 저장시 파일 생성 시간이 변경된다. 따라서 로그 메시지 발생 시간과 파일 생성시간을 확인하여 파일 조작 여부를 판단하는 것이 가능하다.

Ⅵ. 결론 및 향후 계획

침해사고 및 디지털 포렌식 조사에서 파일의 시간정보는 중요한 의미를 가지며, 사용자 및 시스템의 여러 가지 행위에 따라 파일의 시간은 다양하게 변경된다. 디지털 포렌식 분석시 전체 시스템의 타임라인 분석을 통해 사건의 대한 전체 정황과 다양한 증거 획득이 가능하다.

본 논문에서는 리눅스 운영체제에서 파일 조작에 따른 시간 변화의 분석 방안을 제시하였다. 리눅스 Ext4는 파일별로 접근 시간, 수정 시간, Inode 변경 시간, 삭제 시간, 생성 시간의 5가지 시간정보를 저장하는 것을 확인하였다. 이중 접근 시간, Inode 변경 시간, 수정 시간은 일반 사용자가 별도 파일시스템의 지식 없이 어렵지 않게 확인 및 변경이 가능하지만, 파일 생성 시간은 확인 및 변경에 어려움이 따른다.

이를 이용하여 행위에 따른 파일/폴더, 폴더 내부의 파일까지 시간의 변화를 분석하고, 침해사고 및 디지털 포렌식 조사에 활용하는 방안을 연구하였다. 그 결과 악성코드 감염에 대한 침해사고 조사시, 악성코드가 자신의 시간을 변경하더라도 파일 생성 시간을 통해 실제 감염 시간을 획득하는 방안을 확인하였다. 그리고 시스템 로그파일 변조시 파일의 메시지 로그 시간과 생성 시간의 비교를 통해 파일의 조작 여부를 판단 가능하다는 것을 확인하였다. 이는 침해사고와 디지털 포렌식 조사에서 중요한 증거로 활용 될 수 있다.

이러한 연구결과를 통해 향후에는 CentOS, Ubuntu 외에 다양한 버전의 리눅스 운영체제를 대상으로 행위에 따른 파일 및 폴더의 시간 정보 변화를 분석할 예정이다. 그리고 실제 조사에서 활용 가능한 방안을 추가적으로 연구할 계획이다.

References

[1] Jewan Bang, Byeongyeong Yoo, Sangjin Lee, "Analysis of changes in file time attributes with file manipulation", Digital Investigation, Vol. 7, Issues 3-4, pp.135-144, 2011.

[2] Jewan Bang, Byeongyeong Yoo, Jongsung Kim, Sangjin Lee, "Analysis of Time Information for Digital Investigation" INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on, pp.1858-1864, 2009.

[3] Val Henson, Zach Brown, Theodore Ts'o, and Arjan van de Ven, "Reducing fsck time for ext2 file systems," Proceeding of the Linux Symposium, Vol. 1, 2006.

[4] Philip Craiger, "Recovering Digital Evidence from Linux Systems," IFIP The International Federation for Information Processing, Vol. 194, pp.233-244, 2005.

[5] SANS Information, Network, Computer Security Training, Research, Resources, http://www.sans.org.

[6] Hal Pomeranz, "EXT3 File Recovery via Indirect Blocks," http://computer-forensics.sans.org/summit -archives/2011/EXT3-file-recovery.pdf.

[7] Gregorio Narváez, "Taking advantage of Ext3 journaling file system in a forensic investigation," SANS Institute Reading Room, 2007.

[8] Kevin D. Fairbanks, "An analysis of Ext4 for digital forensics," Digital Investigation, Vol. 9, pp. 118-130, 2012.

[9] Dohyun Kim, Jungheum Park, Keun-gi Lee, and Sangjin Lee, "Forensic Analysis of Android Phone using Ext4 File System Journal Log," Lecture Notes in Electrical Engineering, Vol. 164, pp.435-446, 2012.

[10] Dohyun Kim, Jungheum Park, Sangjin Lee, "File Carving for Ext4 File System on Android OS", Journal of The Korea Institute of Information Security & Cryptology(JKIISC), Vol.23, No.3, 2013.

[11] Soeui Kim, Duri Choi, Beongku An, "Detection and Prevention Method by Analyzing Malignant Code of Malignant Bot", The Journal of The Institute of Internet, Broadcasting and Communication(JIIBC), Vol.8, No.2, pp.199-207, 2013.

[12] Operation Windigo Analysis Report, http://www.welivesecurity.com/wp-content/uploads/2014/03/operation_windigo.pdf.

[13] Se-Ryoung Kim, Huy-Kang Kim, "Fuzzy Expert System for Detecting Anti-Forensic Activities", Journal of Internet Computing and Services, Volume 12, Issue 5, pp.47-61, 2011

저자 소개

유 병 영(정회원)

- 2009년 : 상명대학교 컴퓨터과학과 이학사
- 2011년 : 고려대학교 정보경영공학전문대학원 정보보호학과 공학석사
- 2015년~현재 : 국방과학연구소 연구원